TECHNISCHE
UNIVERSITÄT
DARMSTADT

Überwachung und Steuerung dienstbasierter Architekturen – Verteilungsstrategien und deren Umsetzung

Vom Fachbereich Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertation

von

Dipl.-Wirtsch.-Inform. Nicolas Repp

Geboren am 2. Dezember 1977
in Lich, Hessen

Vorsitz: Prof. Dr.-Ing. Peter Meissner
Referent: Prof. Dr.-Ing. Ralf Steinmetz
Korreferent: Prof. Dr.-Ing. Dr. h.c. Peter C. Lockemann

Tag der Einreichung: 26. Mai 2009
Tag der Disputation: 20. Juli 2009

Darmstadt 2009
Hochschulkennziffer D17

ABSTRACT

The application of the SOA paradigm to an enterprise's business processes and the corresponding IT systems allows the realization of business functionalities by the composition and reuse of services. Here, both internal and third party services can be used. Especially in cross-organizational settings, it is crucial to ensure a high level of service quality not only for single services but for service compositions as well. For this, requirements on business processes and systems describing the service levels needed have to be specified and monitored during system runtime. Therefore, appropriate monitoring mechanisms have to be in place, which allow the gathering of data for decision making as well as the execution of countermeasures in case of deviations from requirements specified before. As service-based systems are often distributed and use services across enterprise boundaries, the monitoring and alignment functionalities themselves need to be distributed to ensure scalability, performance and short reaction times to deviations. Furthermore, purely centralized systems will not always have access to all of the data needed for decision making as service providers and intermediaries are often in different control spheres with only limited visibility of monitoring data.

In this thesis, several contributions to the distribution of monitoring and alignment units in service-based infrastructures are made. The main contribution is the development and evaluation of efficient distribution strategies, allowing the computation of distribution schemes by autonomous monitoring and alignment units with limited resources using service level agreement data as well as monitoring data. Therefore, the so called *Monitoring Unit Location Problem* is formulated, which is later on solved by mathematical optimization as well as by the application of heuristics.

Additional contributions address different methodological, architectural, and technological challenges with respect to the distributed monitoring and alignment of services. As a foundation for the distribution of monitoring and alignment units, appropriate modeling of requirements as well as reactions to deviations is needed. Here, a specification language named *Web Service Requirements and Reactions Policy Language* is developed as current modeling approaches only support the specification of requirements. In order to support the implementation of distributed monitoring and alignment in existing service-based infrastructures, appropriate framework support is given. The *Automated Monitoring and Alignment of Services* framework provides an architectural blueprint for the implementation and deployment of distributed monitoring and alignment based on software agent and Web service technologies. Additionally, the framework offers procedures as well as a process model to support the creation of monitored business process instances.

KURZFASSUNG

Die Anwendung des Paradigmas dienstbasierter Architekturen auf die Geschäftsprozesse eines Unternehmens sowie die sie unterstützenden IT-Systeme ermöglicht die Realisierung benötigter Funktionalitäten durch Kombination und Wiederverwendung von Diensten unterschiedlicher Anbieter. Hierbei ist die Sicherstellung eines hohen Dienstgüteniveaus einer Dienstkomposition von hoher Relevanz. Zu dessen Gewährleistung gilt es, die Anforderungen an Geschäftsprozesse und unterstützende IT-Systeme zu spezifizieren sowie deren Einhaltung im laufenden Betrieb zu überwachen. Grundlage hierfür ist die Etablierung geeigneter Überwachungsmechanismen, welche Daten zur Entscheidungsfindung zur Verfügung stellen und Korrekturmaßnahmen bei Nichteinhaltung von Anforderungen initiieren. Die Überwachungs- und Steuerungsfunktionen innerhalb eines verteilten und unternehmensübergreifenden Szenarios sollten aus Gründen der Skalierbarkeit, zur Gewährleistung hoher Performanz und schneller Reaktion auf Anforderungsverletzungen sowie zur Abbildung der unterschiedlichen Datenhoheit und Sichtbarkeit zwischen den beteiligten Parteien ebenfalls verteilt werden.

Die vorliegende Arbeit liefert verschiedene Beiträge zur Verteilung von Überwachungs- und Steuerungseinheiten innerhalb einer dienstbasierten Infrastruktur. Kernbeitrag der Arbeit ist die Entwicklung und Evaluierung von Strategien zur effizienten Verteilung von Einheiten mit der Zielsetzung, die komplexe und rechenintensive Ermittlung von Verteilungen so zu vereinfachen, dass diese durch ressourcenbeschränkte, autonom agierende Überwachungs- und Steuerungseinheiten selbstständig berechnet werden können. Hierbei wird auf vorhandene Leistungsvereinbarungen sowie aktuelle Messdaten zurückgegriffen. Zur Lösung des Verteilungsproblems, dem sogenannten *Monitoring Unit Location Problem*, kommen Verfahren der mathematischen Optimierung sowie Heuristiken zum Einsatz.

Weitere Beiträge der Arbeit adressieren die methodischen, architektonischen und technologischen Herausforderungen einer verteilten Überwachung und Steuerung von Diensten. Grundlegend für die Verteilbarkeit von Überwachungs- und Steuerungseinheiten ist die Möglichkeit, Anforderungen und Reaktionen auf Anforderungsverletzungen geeignet zu modellieren. Hierzu wird die Beschreibungssprache *Web Service Requirements and Reactions Policy Language* entwickelt, welche bisherige Ansätze zur Spezifikation von Dienstgüteanforderungen um Optionen zur Korrektur von Abweichungen erweitert. Unterstützung bei der Realisierung der verteilten Überwachung und Steuerung von Diensten in bestehenden dienstbasierten Infrastrukturen bietet das im Rahmen der Arbeit entwickelte *Automated Monitoring and Alignment of Services* Framework, welches neben einem auf der Webservice- und Softwareagenten-Technologie basierenden Architekturentwurf Prozesse und Vorgehensweisen zur teilautomatisierten Erzeugung überwachter Geschäftsprozessinstanzen zur Verfügung stellt.

Luck, that's when preparation and opportunity meet.

— Pierre Elliott Trudeau (1919-2000)

DANKSAGUNG

Auch wenn am Ende eines Promotionsverfahrens nur eine Person promoviert wird, so ist dies doch der Verdienst von vielen Personen. An dieser Stelle möchte ich mich bei allen bedanken, die direkt oder indirekt an meinem Projekt „Promotion" beteiligt waren.

Sehr herzlich danken möchte ich zunächst meinem Mentor, Arbeitgeber und Betreuer dieser Arbeit Herrn Prof. Dr.-Ing. Ralf Steinmetz für die jahrelange hervorragende und konstruktive Zusammenarbeit sowie die vielen Chancen und Möglichkeiten, die er mir im Rahmen meiner fachlichen Weiterentwicklung geboten hat. Darüber hinaus stand er mir immer mit gutem Rat auch bei schwierigen Entscheidungen zur Seite.

Ebenfalls ganz herzlich bedanken möchte ich mich bei Herrn Prof. Dr.-Ing. Dr. h.c. Peter C. Lockemann für die Übernahme des Zweitgutachtens sowie für das unschätzbare Feedback während der Fertigstellung dieser Arbeit.

Darüber hinaus möchte ich mich bei allen Kolleginnen und Kollegen am Fachgebiet Multimedia Kommunikation für die hervorragende Zusammenarbeit bedanken. Ohne ein solches Team wäre meine Arbeit in dieser Form sicherlich nicht möglich gewesen. Im Besonderen möchte ich mich bei den Mitgliedern der Forschungsgruppe IT-Architekturen Julian Eckert, Stefan Schulte, Michael Niemann, Aneta Kabzeva, André Miede, Apostolos Papageorgiou und Dieter Schuller sowie meinen ehemaligen Kollegen Rainer Berbner, Oliver Heckmann, Andreas Mauthe und Alejandro Perez bedanken. Die langjährige freundschaftliche Zusammenarbeit im Rahmen unterschiedlicher Projekte sowie die Unterstützung in der „heißen" Phase der Promotion werde ich euch nie vergessen! Besonders hervorheben möchte ich an dieser Stelle nochmals Julian, Stefan und Michael, die mir in den letzten Monaten der Entstehung dieser Arbeit große Teile der Projektleitungsarbeit abgenommen und mir somit die notwendigen Freiräume für meine Arbeit geschaffen haben. Gleichermaßen danken möchte ich André für seinen umfangreichen LaTeX-Support.

Bedanken möchte ich mich auch bei allen von mir betreuten Bachelor-, Studien-, Diplom- und Masterarbeitern. Besonders danken möchte ich hierbei Melanie Siebenhaar, die mich auch nach ihrer Studienarbeit als studentische Mitarbeiterin bei meiner Forschungsarbeit unterstützt hat.

Ebenso möchte ich allen Kolleginnen und Kollegen des E-Finance Lab Frankfurt am Main e.V., des Forschungsvorhabens Theseus TEXO sowie aller weiteren Forschungsprojekte für die gute langjährige Zusammenarbeit danken.

Besonderer Dank gebührt meinen Eltern, die mir mein Studium ermöglicht und mich die ganzen Jahre in vielerlei Hinsicht unterstützt haben.

Weiterhin möchte ich mich bei Dir, Bianca, für die Unterstützung während der Promotion bedanken – ohne dein Verständnis und die Inkaufnahme eines häufig arbeitenden Ehemanns wäre meine Promotion nicht möglich gewesen.

Abschließend sei allen Korrekturlesern gedankt für die Zeit und Mühe, die sie in die Durchsicht dieser Arbeit investiert haben.

Darmstadt 2009

INHALTSVERZEICHNIS

1 EINFÜHRUNG

1.1 MOTIVATION UND ZIELSETZUNG

Die zunehmende Globalisierung fordert von Unternehmen verschiedenster Branchen eine immer größer werdende Flexibilität und Agilität. Es reicht nicht mehr aus, auf einem nationalen Markt mit einer etablierten Produktpalette zu agieren – vielmehr müssen Produkte weltweit angeboten werden können. Dem so wachsenden Markt stehen die zunehmende Konkurrenz auf dem Weltmarkt sowie das sich schneller wechselnde Nachfrageverhalten von Konsumenten gegenüber, die eine Anpassung oder Neuentwicklung von Produkten notwendig machen. Um in einem solchen Umfeld bestehen zu können, schließen sich Unternehmen zu strategischen Partnerschaften, Joint Ventures oder innerhalb von globalen Wertschöpfungsketten sowohl horizontal als auch vertikal zusammen (vgl. z. B. [BWW03]). Die Integration der beteiligten Partner bei einer solchen Kooperation kann von einer verhältnismäßig starken Bindung, z. B. im Falle einer Integration von Zulieferunternehmen innerhalb einer Wertschöpfungskette, bis hin zu einer eher dynamischen Partnerschaft variieren, in der nur bedarfsweise, z. B. bei Nachfrageengpässen eines Partners, kooperiert wird.

Gleich welche der beschriebenen Integrationsformen Unternehmen innerhalb einer gemeinsamen Zusammenarbeit wählen – die Geschäftsprozesse der Kooperationspartner müssen entsprechend der neuen aus der Kooperation resultierenden Anforderungen angepasst werden. Je nach zu Grunde liegender Bindung zwischen den Partnern müssen die betroffenen Geschäftsprozesse unterschiedlich stark anpassbar sein, um sich ändernden fachlichen Anforderungen schnell gerecht werden zu können. Allerdings reicht die Anpassung der Geschäftsprozesse auf fachlicher Ebene nicht aus, um wechselnde Anforderungen abbilden und auf diese Weise zum Geschäftserfolg der Unternehmung beitragen zu können. Im Zeitalter der elektronischen Informationsverarbeitung sind je nach Branche bis zu 80 Prozent der Geschäftsprozesse eines Unternehmens durch Informationstechnologien (IT) unterstützt oder vollständig durch diese abgebildet [SL08], [BWK05].[1] Eine Änderung von Geschäftsprozessen bedingt somit in den meisten Fällen auch eine Änderung bestehender IT-Systeme oder sogar deren partielle Neuentwicklung. Gleiches gilt für die Abbildung einer Kooperation zwischen Unternehmen auf die bestehende Systeminfrastruktur und -architektur. Zur Unterstützung der Kooperation müssen existierende IT-Systeme der Kooperationspartner über Unternehmensgrenzen hinweg zusammengeschlossen werden. Hierbei gilt es zu berücksichtigen, dass die bestehenden Systeme häufig heterogen sind und nicht ohne weiteres für eine solche Öffnung über Unternehmensgrenzen hinweg ausgelegt sind.

1 Konkret ergab eine Studie unter kleinen und mittleren Unternehmen aus dem Jahr 2008, dass 25 Prozent der Unternehmen IT innerhalb eines Bereichs einsetzen, 30 Prozent dies bereichsübergreifend tun (häufig im Vertrieb und der Produktion) und 25 Prozent IT sogar bereits zur Realisierung unternehmensübergreifender Prozesse nutzen (speziell im Supply Chain Management und im Finanzbereich) [SL08].

In den letzten Jahren ist im Bereich der Anwendungsintegration (engl. Enterprise Application Integration – EAI) die Webservice-Technologie aufgekommen, welche Unterstützung bei der Integration von Anwendungssystemen innerhalb von Unternehmen und über Unternehmensgrenzen hinweg verspricht. Webservices und die sie unterstützenden Standards bauen auf der etablierten Infrastruktur des Internets sowie deren zu Grunde liegenden Protokollen und Standards auf, wie z. B. dem Hypertext Transfer Protocol (HTTP – vgl. [FGM+99]) als etabliertem Transportmechanismus, und ergänzen diese um Fähigkeiten zum Fernaufruf von Funktionen über klar definierte und webbasierte Schnittstellen hinweg. Hierbei wird zumeist Gebrauch von der Extensible Markup Language (XML) gemacht, um die zu übertragenden Daten in einem maschinenlesbaren Format zu beschreiben. In Abgrenzung zu wesentlich länger existierenden Ansätzen, wie z. B. den Remote Procedure Calls (RPC) oder der Common Object Request Broker Architecture (CORBA), bauen sie ausschließlich auf der Verwendung offener Standards auf.

Mittlerweile besteht ein Hauptanwendungsfeld der Webservice-Technologie in der Realisierung von IT-Systemen, welche auf dem Paradigma der dienstbasierten Architekturen (engl. Service-oriented Architectures – SOA) basieren. Das SOA-Paradigma, welches unter dieser Bezeichnung erstmals im Jahre 1996 durch die Gartner Group beschrieben wurde [SN96], [Sch96], definiert ein Konzept, mit dessen Hilfe komplexe Informationssysteme entworfen und strukturiert werden können. Hintergrund für die Definition des SOA-Paradigmas war die Beobachtung, dass eine getrennte Betrachtung von IT und den Anforderungen der Fachabteilungen zu einer Vielzahl von Problemen, wie beispielsweise mangelnde Flexibilität existierender Geschäftsprozesse oder schlechte Reaktionszeiten bei der Umsetzung neuer Produkte, führen kann. Um dieses Problem adäquat adressieren zu können, wird zur Integration von IT mit fachlichen Anforderungen im Rahmen des Paradigmas der dienstbasierten Architekturen die Abstraktionsschicht der „Dienste“ definiert, die sowohl für die fachliche als auch die IT-Seite eine gemeinsame Bedeutung besitzt bzw. deren Bedeutungen sich ineinander überführen lassen. Bei der konsequenten Anwendungen des Paradigmas der dienstbasierten Architekturen auf ein Unternehmen, dessen Geschäftsprozesse sowie dessen IT-Systeme, lassen sich die Geschäftsprozesse und deren Bestandteile (Teilprozesse sowie einzelne Tätigkeiten) auf verschiedene unterschiedlich gestaltete Dienste herunterbrechen, welche die benötigte Funktionalität zur Verfügung stellen können. Geeignete Dienste können hierbei rein technische Dienste sein, wie z. B. einzelne Webservices, oder aber auch durch Mitarbeiter und Organisationseinheiten erbrachte Leistungen.

Von besonderer Relevanz innerhalb eines Geschäftsszenarios ist die Qualitätssicherung der durch die Geschäftsprozesse sowie der sie unterstützenden IT-Systeme erbrachten Leistungen. Die Qualität kann hierbei auf verschiedenen Ebenen betrachtet werden. Eine wichtige Rolle nimmt die zumeist subjektive Beurteilung eines Geschäftsprozesses und der daran beteiligten IT-Systeme durch deren Nutzer ein, welche sich auch in den Erfahrungen der Nutzer widerspiegelt (engl. Quality of Experience – QoE). Gleichermaßen ist die Dienstgüte (engl. Quality of Service – QoS) der Geschäftsprozesse sowie der unterliegenden IT-Systeme relevant, wobei man unter dem Begriff der Dienstgüte beispielsweise die Bearbeitungsdauer eines Geschäftsprozesses, die Verfügbarkeit der ihn unterstützenden IT-Systeme oder die Fehleranfälligkeit bei der Prozessbearbeitung versteht. Die Sicherstellung eines hohen Dienstgüte-

niveaus innerhalb eines dienstbasierten Systems stellt den Betrachtungsgegenstand dieser Arbeit dar.

In der Praxis ist häufig eine Lücke zwischen dem durch einen Nutzer erwarteten Qualitätsniveau, den durch eine Organisation spezifizierten Dienstgüteanforderungen und der tatsächlich erbrachten Dienstgüte zu beobachten, die sich auf eine Vielzahl von Ursachen zurückführen lässt. Im Falle dienstbasierter Geschäftsprozesse zählen hierzu neben den rein fachlichen Ursachen, wie der unzutreffenden Spezifikation von Anforderungen und übersteigerter Erwartungen an das Qualitätsniveau, verschiedene technische Einflussfaktoren. Da es sich bei einem solchen dienstbasierten System um ein verteiltes System handelt, welches auf Grundlage von Internetstandards über Netzwerke hinweg miteinander kommuniziert, lassen sich auf allen Ebenen eines Protokollstapels Einflussfaktoren identifizieren, wie z. B. die Überlast von Netzkomponenten oder die Nichtverfügbarkeit von Dienstanbietern.

Zur Sicherstellung eines hohen Qualitätsniveaus im Hinblick auf die Dienstgüte gilt es, ein geeignetes Dienstgütemanagement einzuführen. Voraussetzungen für das Management von Dienstgüte sind die Spezifikation der Anforderungen an die jeweiligen Geschäftsprozesse sowie die Ableitung der zugehörigen Anforderungen an die unterliegenden IT-Systeme. Entsprechende Anforderungen können beispielsweise im Rahmen von Verträgen und Leistungsvereinbarungen (engl. Service Level Agreements – SLA) vorgenommen und somit für die involvierten Parteien verbindlich festgeschrieben werden.

Allerdings reicht die reine Spezifikation von Anforderungen in Bezug auf die Dienstgüte von Geschäftsprozessen und IT-Systemen nicht aus. Die Einhaltung entsprechender Anforderungen im laufenden Betrieb muss zu jedem Zeitpunkt sichergestellt werden können. Hierzu ist es notwendig, geeignete Überwachungsmechanismen und Systeme zu etablieren, welche die notwendigen Daten generieren oder auslesen, aufbereiten und zur weiteren Entscheidungsfindung zur Verfügung stellen. Entsprechende Gegenmaßnahmen im Falle der Nichteinhaltung von Anforderungen können die Übergabe des Falls an eine manuelle Nachbearbeitung ebenso sein, wie die automatisierte Durchführung von Gegenmaßnahmen durch das IT-System selbst.

Großen Einfluss auf die Überwachung der Dienstgüte von Geschäftsprozessen und den zugehörigen IT-Systemen besitzt die Gestaltung der Informationssystemarchitektur. Informationssysteme auf Basis dienstbasierter Architekturen, wie sie dieser Arbeit zu Grunde liegen, bieten die Möglichkeit der Verteilung benötigter Funktionalitäten auf unterschiedliche Parteien durch die Einbindung von Diensten unternehmensinterner und -externer Anbieter. Gerade die Integration von Diensten über Unternehmensgrenzen hinweg stellt aber eine besondere Herausforderung für die Überwachung und Sicherstellung von Dienstgüteanforderungen dar. Die einzubindenden Dienste befinden sich in fremden Kontrollsphären, in denen kein oder nur eingeschränkter Zugriff auf für das Dienstgütemanagement notwendige Informationen möglich ist.

Klassische Verfahren aus dem Bereich des Netzwerkmanagements sowie die den aktuellen SOA-Plattformen zu Grunde liegenden Verfahren greifen zur Überwachung von Diensten auf eine zentral organisierte Architektur zurück, in welcher Informationen über den Zustand eines Dienstes zwar verteilt gesammelt, jedoch zentral aggregiert und ausgewertet werden. Die Überprüfung auf die Einhaltung von Dienstgüteanforderungen sowie die Steuerung möglicher Maßnahmen im Falle von

Anforderungsverletzungen wird im Falle bestehender SOA-Lösungen zumeist durch den Dienstaufrufenden selbst durchgeführt.

Dass eine solche zentrale Überwachung und Steuerung nicht in allen Fällen zur zeitnahen Reaktion auf Anforderungsverletzungen ausreicht, haben Untersuchungen am Beispiel der Webservice-Technologie als gängige Form der Implementierung von SOA-Diensten gezeigt. Experimente mit Webservices innerhalb eines Testbeds sowie mit frei im Internet verfügbaren Webservices haben gezeigt, dass beispielsweise die Detektion eines Dienstausfalls durch die zentrale Instanz je nach Implementierung mehrere Minuten benötigen kann [RBHS07], [RBE+07]. Erst nach dieser Zeit können evtl. notwendige Gegenmaßnahmen eingeleitet werden, um die Erfüllung von Dienstgüteanforderungen zu gewährleisten. Weiterhin sind bei einer ausschließlich zentralen Überwachung und Steuerung zumeist nicht alle notwendigen Daten verfügbar, da die Eigner der verschiedenen Kontrollsphären diese aus Gründen der Vertraulichkeit nicht oder nur eingeschränkt zur Verfügung stellen.

Ausgehend von diesen Beobachtungen wird im Rahmen der vorliegenden Arbeit die Strategie verfolgt, die notwendigen Überwachungs- und Steuerungsfunktionalitäten für ein solches dienstbasiertes Szenario analog zu den Diensten selbst innerhalb der zu Grunde liegenden Infrastruktur zu verteilen. Hierbei wird eine Verteilung in fremden Kontrollsphären ebenso unterstützt, wie die Platzierung von Überwachungs- und Steuerungseinheiten in der unternehmenseigenen Infrastruktur. In diesem Kontext können fremde Kontrollsphären sowohl den Einflussbereich der Dienstanbieter selbst, als auch die Einflussbereiche von Intermediären, wie beispielsweise Netzwerkanbietern, kennzeichnen. Je näher eine solche Überwachung und Steuerung von Diensten aus topologischer Sicht am Ort der eigentlichen Dienstausführung stattfindet, desto früher ist die Erkennung von Anforderungsverletzungen sowie die Einleitung von Gegenmaßnahmen möglich. Die Platzierung von autonom agierenden Überwachungs- und Steuerungseinheiten innerhalb fremder Kontrollsphären ermöglicht weiterhin, dass vertrauenswürdige Daten verwendet werden können, ohne dass die Grenzen der Kontrollsphären bzw. die Datenhoheit verletzt werden. Darüber hinaus lässt sich durch eine solche Verteilung der aus Überwachungs- und Steuerungsmaßnahmen resultierende Datenverkehr reduzieren. Weitere Nebeneffekte der Verteilung von Überwachungs- und Steuerungsfunktionalitäten sind die Verbesserung der Skalierbarkeit der dienstbasierten Lösung sowie die Steigerung der Robustheit durch Einführung unabhängiger Überwachungs- und Steuerungseinheiten, die füreinander Aufgaben übernehmen.

Die Adressierung ausgewählter Herausforderungen der Verteilung von Überwachungs- und Steuerungseinheiten innerhalb einer dienstbasierten Infrastruktur ist Gegenstand dieser Arbeit. Der Schwerpunkt der Arbeit liegt hierbei auf der Entwicklung und Evaluierung von Strategien zur effizienten Verteilung der Überwachungs- und Steuerungseinheiten in einer dienstbasierten Infrastruktur. Ziel ist es, die komplexe und rechenintensive Ermittlung von Verteilungen so zu vereinfachen, dass diese durch ressourcenbeschränkte, autonom agierende Überwachungs- und Steuerungseinheiten selbstständig ermittelt werden können. Die entwickelten Lösungsverfahren können auf Basis von Informationen aus vorhandenen Leistungsvereinbarungen sowie zur Laufzeit gewonnenen Messdaten eine kostenminimierende Verteilung vornehmen und diese in sich ändernden Umgebungen kontinuierlich anpassen.

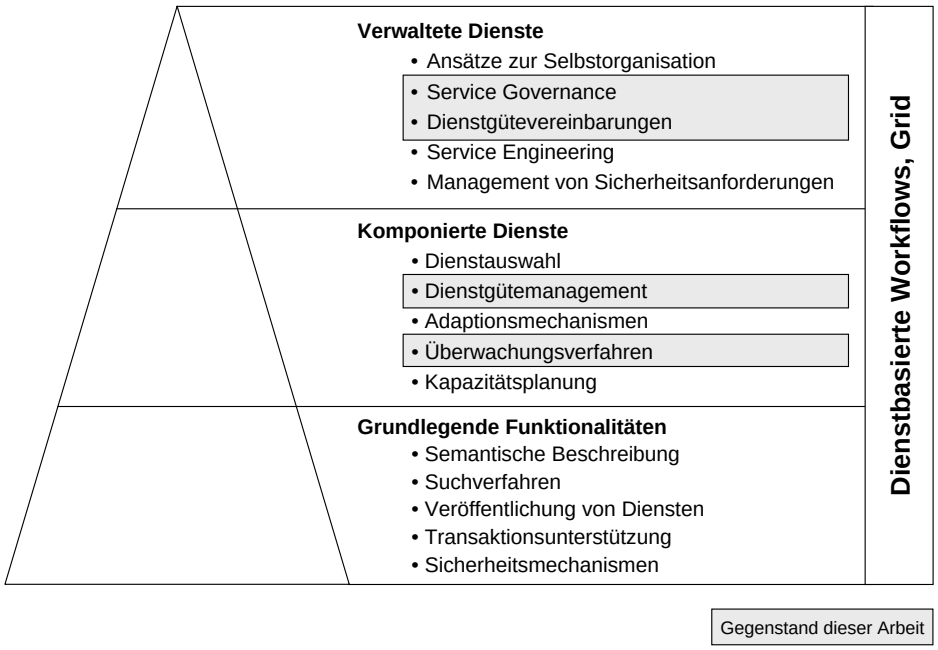


Abbildung 1: Forschungsagenda für dienstbasierte Architekturen (nach [Pap03], [Ber08])

Weiterhin beschäftigt sich die Arbeit mit den methodischen, architektonischen und technologischen Herausforderungen, die bei der Umsetzung einer verteilten Überwachung und Steuerung innerhalb einer dienstbasierten Infrastruktur auftreten.

1.2 BEITRÄGE DIESER ARBEIT

Im Rahmen der vorliegenden Arbeit werden im Detail unterschiedliche, für die verteilte Überwachung und Steuerung von Diensten notwendige Aspekte untersucht. Die Sicherstellung einer hohen Dienstgüte von dienstbasierten Geschäftsprozessen wird hierbei aus Sicht einer möglichen Unterstützung durch geeignete Infrastrukturkomponenten betrachtet. Eine Einordnung der Beiträge dieser Arbeit in eine übergeordnete Forschungsagenda für dienstbasierte Architekturen in Anlehnung an [Pap03] und [Ber08] wird in Abbildung 1 dargestellt.

Die Beiträge der vorliegenden Arbeit zur aktuellen Forschung im Bereich dienstbasierter Architekturen lassen sich in die folgenden Teilbeiträge untergliedern:

1. Entwicklung und Evaluierung von Strategien zur effizienten Verteilung von Überwachungs- und Steuerungseinheiten in dienstbasierten Infrastrukturen.
2. Entwicklung und prototypische Umsetzung von Hilfsmitteln für die verteilte Überwachung und Steuerung in dienstbasierten Infrastrukturen:
   a) Entwicklung einer Sprache zur integrierten Spezifikation von Anforderungen an Dienste sowie der Reaktionen auf Abweichungen.
   b) Entwicklung eines Frameworks zur methodischen, architektonischen und technologischen Unterstützung der Umsetzung.

Nachfolgend werden die Teilbeiträge im Detail vorgestellt.

Den Kern der vorliegenden Arbeit stellt ein Beitrag zur Planung der Überwachung dienstbasierter Infrastrukturen dar, in dem untersucht wird, an welchen Lokationen und innerhalb welcher Kontrollsphären Überwachungs- und Steuerungseinheiten zur Optimierung des Reaktionsverhaltens bei Anforderungsverletzungen sowie der Reduktion des Datenaufkommens zu platzieren sind (Teilbeitrag 1). Zur Ermittlung der Verteilung kommen Verfahren der mathematischen Optimierung zum Einsatz, welche auf den Inhalten von Dienstgütevereinbarungen als auch aktuellen Messwerten operieren. Diese können sowohl zur Entwurfszeit einer dienstbasierten Infrastruktur als auch zur späteren Laufzeit des Systems, welches die Infrastruktur nutzt, ausgeführt werden. Grundlage für die Optimierung stellt das im Rahmen der Arbeit entwickelte und als *Monitoring Unit Location Problem* (MULP) bezeichnete Verteilungsproblem für Überwachungs- und Steuerungseinheiten dar. Die Analyse verwandter Problemstellungen (vgl. z. B. [DD95] und [DK97]) hat gezeigt, dass vergleichbare Optimierungsprobleme NP-schwer sind und auf Grund des hohen Berechnungsaufwands eine Lösung mit exakten Verfahren in realistischen Szenarien nicht in annähernd Echtzeit durch ressourcenbeschränkte Einheiten ermittelt werden kann. Zur Adressierung dieser Herausforderungen werden im Rahmen der Arbeit heuristische Lösungsstrategien entwickelt und umgesetzt. Die Eignung der Strategien wird durch eine simulationsbasierte Evaluierung belegt.

Zur Umsetzung der zuvor diskutierten Verteilungsstrategien innerhalb einer dienstbasierten Infrastruktur wird ein geeignetes Überwachungs- und Steuerungssystem benötigt, zu dessen Gestaltung die Arbeit umfangreiche methodische, architektonische und technologische Unterstützung liefert (Teilbeitrag 2).

Grundlegend für die Verteilung von Überwachungseinheiten und im Besonderen für die Verteilung der Steuerungs- bzw. Korrekturlogik ist die Möglichkeit, Anforderungen und Reaktionen auf Abweichungen von definierten Anforderungen geeignet modellieren zu können. Im Rahmen der Arbeit wird für diesen Anwendungsfall eine Spezifikationssprache entworfen, mit deren Hilfe gleichzeitig Anforderungen und Reaktionen auf Abweichungen definiert werden können (Teilbeitrag 2a). Die *Web Service Requirements and Reactions Policy Language* (WS-Re2Policy) adressiert hiermit ein Manko bisheriger Ansätze zur Spezifikation von Dienstgüteanforderungen, die zumeist nur auf die Anforderungen selbst und weniger auf die möglichen Optionen zur Korrektur von Abweichungen eingehen. Gerade letzteres ist aber notwendig, um die Handlungsspielräume teilautonom agierender Überwachungs- und Steuerungseinheiten ausreichend zu definieren. Die entwickelte WS-Re2Policy-Sprache baut hierbei auf der WS-Policy-Sprache des World Wide Web Consortiums (W3C) auf. Sie lässt sich durch alle zu WS-Policy kompatiblen Sprachen zur Spezifikation von Anforderungen erweitern.

Die Realisierung eines verteilten Überwachungs- und Steuerungssystems sowie die Integration der bisher beschriebenen Beiträge werden durch das im Rahmen der Arbeit entwickelte *Automated Monitoring and Alignment of Services* (AMAS.KOM) Framework adressiert (Teilbeitrag 2b). Grundlage des Frameworks ist ein Architekturentwurf, der auf der Anwendung ausgewählter Eigenschaften des Paradigmas mobiler Softwareagenten innerhalb eines dienstbasierten Systems basiert. Die Nutzung mobiler Softwareagenten ermöglicht die Definition von Überwachungs- und Steuerungseinheiten im Sinne autonomer mobiler Softwarekomponenten, die es aus technischer Sicht ermöglichen, in verschiedenen Szenarien und an verschie-

denen Lokationen ausgeführt zu werden. Mit ihrer Hilfe ist es möglich, einzelnen Überwachungs- und Steuerungseinheiten die Verantwortlichkeit für Teile der Prozesse zu übergeben und somit den Eingangs definierten Anforderungen nach Skalierbarkeit, Robustheit sowie der Verbesserung der Reaktionszeit aus rein technischer Sicht nachzukommen. In Ergänzung zur Architektur bietet das Framework ein operationalisierbares Vorgehensmodell für die Erzeugung überwachter Geschäftsprozessinstanzen auf Dienstbasis. Als Bestandteil des Prozesses zu Generierung einer überwachten Prozessinstanz wird ein teilautomatisierter Transformationsprozess entwickelt, welcher eine Prozessbeschreibung auf Basis des De-Facto-Standards *Web Service Business Process Execution Language* (WSBPEL) mit zugehöriger Spezifikation von Anforderungen und Reaktionen auf Abweichungen in WS-Re2Policy in eine überwachte Prozessinstanz überführt und zur kontrollierten Ausführung bringt. Integraler Bestandteil des Frameworks ist hierbei die automatisierte Generierung von Überwachungs- und Steuerungseinheiten. Eine möglicherweise notwendige Partitionierung von Geschäftsprozessen in Teilprozesse und einzelne Aufgaben sowie deren Zuordnung zu Diensten, welche deren Funktionalität abbilden, sind nicht Gegenstand dieser Arbeit. Hierfür kommen Kompositionsverfahren für Dienste zum Einsatz, wie sie z. B. in [BSR+06b] oder [CPEV08] erläutert werden.

Als Nachweis für die Realisierbarkeit des durch die verschiedenen Beiträge dieser Arbeit definierten Überwachungs- und Steuerungssystems wird dieses prototypisch auf Basis der Softwareagenten- und Webservice-Technologie implementiert und unter verschiedenen Aspekten evaluiert. Insbesondere wird untersucht, welche Performanzveränderungen resultierend aus der Integration zusätzlicher Softwarekomponenten die Einführung eines solchen Systems verursacht.

1.3 AUFBAU DER ARBEIT

Die weitere Arbeit ist wie folgt aufgebaut: In Kapitel 2 wird in die Grundlagen dienstbasierter Architekturen eingeführt und erläutert, wie dienstbasierte Architekturen und Geschäftsprozesse innerhalb eines Unternehmens miteinander verknüpft sind. Weiterhin werden Anwendungsfelder dienstbasierter Architekturen innerhalb eines Unternehmens vorgestellt. Die Herleitung eines Modells zur Beschreibung der Qualität dienstbasierter Systeme ist gleichermaßen Gegenstand des Kapitels. Ergänzend werden das Management von Dienstgüte sowie die Sicherstellung von Dienstgüte durch geeignete Überwachungs- und Steuerungsmechanismen eingeführt und motiviert.

Kapitel 3 beschäftigt sich mit der Verteilung von Überwachungs- und Steuerungseinheiten in einer dienstbasierten Infrastruktur. Innerhalb des Kapitels werden verschiedene Strategien zur Verteilung von aktiven Softwarekomponenten zur Überwachung und Steuerung von Diensten in einer dienstbasierten Infrastruktur entwickelt und untersucht. Hierbei kommen Ansätze aus dem Bereich der gemischtganzzahligen linearen Programmierung sowie eigenentwickelte Heuristiken zum Einsatz. Die Verteilungsstrategien werden im Rahmen von Simulationsexperimenten auf ihre Anwendbarkeit sowie ihre Leistungsmerkmale hin untersucht. Es wird gezeigt, dass die Verteilung von Überwachungs- und Steuerungseinheiten innerhalb

einer dienstbasierten Architektur unter bestimmten Rahmenbedingungen Verbesserungen in der Reaktionszeit auf Fehler und andere Abweichungen von gewünschtem Systemverhalten bringen kann und darüber hinaus die dabei entstehenden Kommunikationskosten im Vergleich zu zentralen Lösungen minimiert. Die Diskussion verwandter Arbeiten schließt Kapitel 3 ab.

Im Rahmen von Kapitel 4 wird untersucht, welche Anforderungen an eine Konfigurationssprache für Überwachungs- und Steuerungseinheiten existieren. Um die Wiederverwendung von Überwachungs- und Steuerungseinheiten zu ermöglichen, wird bei ihrer Entwicklung das Prinzip der „Konfiguration vor Programmierung" verfolgt. In diesem Sinne lassen sich einzelne Überwachungs- und Steuerungseinheiten allein auf Basis einer ihnen übergebenen Konfiguration an ihre jeweiligen Aufgabengebiete anpassen. Die Konfiguration muss hierbei nicht nur Anforderungen an die zu überwachenden Dienstgütemerkmale beinhalten, sondern darüber hinaus die Spezifikation von möglichen Korrekturmaßnahmen unterstützen. Auf Grund der Tatsache, dass die untersuchten existierenden Ansätze zur Konfiguration und Spezifikation nur reine Anforderungsspezifikationssprachen darstellen, wird in dieser Arbeit eine für den vorliegenden Anwendungsfall geeignete Spezifikationssprache entwickelt, die sowohl die Spezifikation von zu überwachenden Anforderungen als auch die Beschreibung durchzuführender Korrekturmaßnahmen ermöglicht. Kapitel 4 schließt ebenfalls mit der Diskussion verwandter Arbeiten.

Nachfolgend wird in Kapitel 5 das in dieser Arbeit entwickelte AMAS.KOM Framework vorgestellt, mit dessen Hilfe die Verteilung von Überwachungs- und Steuerungseinheiten innerhalb einer dienstbasierten Infrastruktur realisiert werden kann. Auf Basis der Ermittlung notwendiger Anforderungen an ein solches Framework wird eine geeignete Architektur entwickelt und nachfolgend prototypisch implementiert. Parallel zum Entwurf des Frameworks werden ein Vorgehensmodell und ein Transformationsprozess entworfen, welche die Grundlage der Erzeugung überwachter Prozessinstanzen darstellen. Diese werden ebenfalls im Rahmen des Kapitels vorgestellt. Das Kapitel schließt mit einer Evaluation des prototypisch umgesetzten AMAS.KOM Frameworks hinsichtlich der durch die Einführung des Frameworks entstehenden Leistungsveränderung des Gesamtsystems sowie der Diskussion verwandter Arbeiten.

Die Arbeit schließt in Kapitel 6 mit der Zusammenfassung der wesentlichen Ergebnisse und gibt einen Ausblick auf Erweiterungsmöglichkeiten und weitere Einsatzgebiete der im Rahmen der Arbeit entwickelten Ansätze.

2 GRUNDLAGEN

In diesem Abschnitt werden die wichtigsten Begrifflichkeiten und Konzepte eingeführt und definiert, welche in der nachfolgenden Arbeit Verwendung finden.

2.1 DIENSTE UND DIENSTBASIERTE ARCHITEKTUREN

2.1.1 *Definitionen und Terminologie*

Dienstbasierte bzw. dienstorientierte Architekturen stellen ein Gestaltungsparadigma für Unternehmens- und IT-Architekturen dar, welches in den letzten Jahren sowohl in der Industrie als auch der Forschung massiv an Bedeutung gewonnen hat [RM08].[2] Allerdings muss man feststellen, dass sich weder in der Industrie noch in der Forschung zum aktuellen Zeitpunkt ein einheitliches Verständnis oder eine einheitliche Definition für dienstbasierte Architekturen durchgesetzt hat. Als Basis zur Entwicklung einer für diese Arbeit gültigen Arbeitsdefinition wird nachfolgend in diesem Abschnitt eine Reihe von Definitionen für dienstbasierte Architekturen vorgestellt. Gemeinsamer Nenner der gängigen Definitionen ist die Betrachtung von SOA als Gestaltungsparadigma nicht ausschließlich als Technologie, Standard oder konkretes Produkt (vgl. z. B. [Jos08]). Die jeweilige Schwerpunktsetzung zwischen der Betrachtung als rein fachlichem Konzept und konkreter Technologie unterscheidet sich allerdings von Autor zu Autor.

Eine der am weitesten verbreiteten Definitionen für dienstbasierte Architekturen ist die im Rahmen des Referenzmodells für dienstbasierte Architekturen des Standardisierungsgremiums OASIS (Organization for the Advancement of Structured Information Standards) aufgeführte Definition. OASIS bezeichnet dienstbasierte Architekturen als „Paradigma zur Organisation und Nutzung verteilter Ressourcen, die von verschiedenen Anbietern zur Verfügung gestellt werden können“ [MLM+06]. Dienste stellen nach der Definition von OASIS den abstrakten Betrachtungsgegenstand dar, der verwendet wird, um die Verbindung zwischen einem Angebot und einer potentiellen Nachfrage zu realisieren. Dienste können in diesem Sinne von Menschen und auch IT-Systemen erbracht werden. Eine Referenz auf Technologien zur Umsetzung von dienstbasierten Architekturen wird in der Definition von OASIS nicht vorgenommen.

Gleichermaßen aus einer fachlichen Sicht heraus entwickelt ist die Definition für dienstbasierte Architekturen bei [Woo04] und [Hum08]. Als SOA wird bei den Autoren ein Paradigma verstanden, welches zur „Strukturierung des Geschäfts eines Unternehmens“ verwendet und auf dessen Basis nachfolgend die zugehörige IT-Architektur entwickelt werden kann. Die Zweiteilung in eine fachliche als auch

2 Im Rahmen dieser Arbeit wird der Begriff „Dienstbasierte Architektur“ synonym zu dessen sinngemäßer englischsprachiger Übersetzung „Service-oriented Architecture“ verwendet. Anstelle einer deutschsprachigen Abkürzung wird im weiteren Verlauf die etablierte Abkürzung „SOA“ genutzt.

technische Sicht wird auch beim Verständnis des Dienstbegriffs beibehalten. Dienste können in „Geschäftsdienste“ und „Anwendungsdienste“ unterschieden werden, wobei ein Geschäftsdienst eine geschäftliche Leistung (Dienstleistung) darstellt, die von einem Dienstanbieter für einen Dienstnutzer erbracht wird [EHH+08], [Hum08]. Dienstnutzer können unternehmensintern als auch -extern sein. Ein Anwendungsdienst ist in diesem Zusammenhang ein Geschäftsdienst, der teilweise oder vollständig durch den Einsatz von IT erbracht wird. Als besonderes Merkmal einer auf Diensten basierenden Architektur wird der Zusammenhang zwischen der fachlichen Sicht auf Dienste sowie deren technischer Unterstützung gesehen. Technische Dienste im Sinne von Anwendungsdiensten können nur dann spezifiziert und umgesetzt werden, wenn die zugehörigen Geschäftsdienste klar definiert sind. Der im Rahmen dieser Arbeit weiterhin verwendete Dienstbegriff bezieht sich, soweit nicht anderes aufgeführt, auf „Anwendungsdienste“.

Ebenfalls eine Konkretisierung des Paradigmas in Richtung eines möglichen Einsatzes im betrieblichen Kontext wird bei [BBF+06] diskutiert. Die Autoren definieren eine dienstbasierte Architektur als „Framework zur Integration von Geschäftsprozessen mit der die Prozesse unterstützenden IT-Infrastruktur“. In diesem Zusammenhang werden Dienste als standardisierte Komponenten betrachtet, mit deren Hilfe eine solche Integration realisiert werden kann, um sich den wandelnden Anforderungen aus dem betrieblichen Kontext anpassen zu können. Weiteres wichtiges Entwurfsmerkmal dienstbasierter Architekturen ist die lose Kopplung zwischen Geschäftsprozesssicht und der unterstützenden IT-Infrastruktur. Gleichermaßen resultiert hieraus die lose Kopplung der zur Abbildung der benötigten Funktionalität kombinierten Dienste.

Eine Definition einer dienstbasierten Architektur, welche das Paradigma wesentlich konkreter im Hinblick auf seine potentielle technologische Umsetzung sieht, ist die Definition von [MES+08]. Die Autoren bezeichnen eine dienstbasierte Architektur als „Systemarchitektur, die vielfältige, verschiedene und eventuell inkompatible Methoden und Applikationen als wiederverwendbare und offene zugreifbare Dienste repräsentiert und dadurch eine plattform- und sprachenübergreifende Nutzung und Wiederverwendung ermöglicht“. Im Vergleich zu den vorangegangenen Definitionen stellt diese Definition eine Verschiebung des Betrachtungsschwerpunkts in Richtung eines eher technisch geprägten Paradigmas dar, auch wenn keine konkret einzusetzenden Technologien aufgeführt werden.

An der technischen Umsetzung orientiert sind auch die Definitionen von [RH06] sowie der Web Services Architecture Arbeitsgruppe des World Wide Web Consortium (W3C) [BHM+04]. [RH06] beschreiben eine dienstbasierte Architektur als Architektur, bei der alle Funktionalitäten als Dienste gekapselt sind, welche über standardisierte und öffentlich zugängliche Schnittstellen verfügen. Entsprechende Dienste müssen lose gekoppelt sein. Ebenfalls sehr techniknah beschreibt das W3C eine dienstbasierte Architektur als nachrichtenbasiertes und verteiltes System, innerhalb dessen Dienste Abstraktionen konkreter Anwendungen, Geschäftsprozesse und auch Datenquellen darstellen. Entsprechende Dienste innerhalb eines solchen verteilten Systems müssen maschinell konsumierbar, d. h. durch ein anderes IT-System auffindbar und nutzbar sein. Darüber hinaus macht die Arbeitsgruppe des W3C konkrete Vorschläge für die Umsetzung einer solchen Architektur und ihrer Dienste, indem z. B. der Einsatz der Extensible Markup Language (XML) zur Beschreibung der Dienste als auch zur Übertragung der Daten empfohlen wird [BHM+04].

Abschließend soll noch die Definition von Erl (vgl. [Erl05]) dargestellt werden, der als Ergänzung zu den vorangegangenen Definitionen noch weitere Eigenschaften einer dienstbasierten Architektur beschreibt. Nach Erl zeichnet eine „zeitgemäße" dienstbasierte Architektur aus, dass sie offen und flexibel ist, leicht zu erweitern ist sowie aus unabhängigen und von verschiedenen Anbietern stammenden Diensten zusammengesetzt werden kann. Ebenfalls kommt der Interoperabilität und Wiederverwendung von Diensten ein hoher Stellenwert zu. Nicht zuletzt sollen entsprechende Dienste einen hohen Grad an Dienstgüte zur Verfügung stellen, d. h. qualitativ hochwertige Funktionen erbringen. Gleichermaßen definiert eine dienstbasierte Architektur im Sinne von Erl eine Abstraktionsschicht zwischen der Geschäftslogik (der fachlichen Domäne) sowie der zum Einsatz kommenden Technologien. Eine zeitgemäße dienstbasierte Architektur ist im Sinne von Erl auf Basis der Webservice-Technologie aufgebaut.

Die vorangegangenen Definitionen bilden die Basis für eine Arbeitsdefinition, die der nachfolgenden Arbeit zu Grund gelegt wird. Im Sinne dieser Arbeit wird als eine dienstbasierte Architektur eine Architektur verstanden, die aus lose gekoppelten Diensten verschiedenster Anbieter zusammengesetzt werden kann. Hierbei stellt ein Dienst eine sich selbst beschreibende Kapselung von Funktionalitäten einer fachlichen Domäne dar [Pap03], [KBS05]. In Anlehnung an [HJ06] und [ABC+02] soll dieser Dienstbegriff weiter präzisiert werden. Dienste können gleichermaßen als Schnittstellen von Komponenten verstanden werden, die nach [ABC+02] wie folgt definiert werden:

„Eine Komponente besteht aus verschiedenartigen (Software-)Artefakten. Sie ist wiederverwendbar, abgeschlossen und vermarktbar, stellt Dienste über wohl definierte Schnittstellen zur Verfügung, verbirgt ihre Realisierung und kann in Kombinationen mit anderen Komponenten eingesetzt werden, die zur Zeit der Entwicklung nicht unbedingt vorhersehbar sind."

Von speziellem Interesse sind im Rahmen dieser Arbeit die Fachkomponenten, d. h. Komponenten, die Funktionalitäten aus einer betrieblichen Anwendungsdomäne abbilden [FRT99]. Handelt es sich bei diesen um Softwareartefakte, so sind die durch die Fachkomponenten bereitgestellten Dienste Anwendungsdienste im Sinne dieser Arbeit, welche sich zu Geschäftsprozessen bzw. zu Workflows, d. h. den durch IT unterstützten Teilen von Geschäftsprozessen, komponieren lassen.

2.1.2 *Betrachtungsaspekte von Diensten*

Auf Basis der in [Erl05] beschriebenen Merkmale dienstbasierter Architekturen, den Entwurfsempfehlungen für SOA in [HJ06], den in [ABC+02] vorgestellten Kriterien für Fachkomponenten sowie der Ergebnisse aus empirischen Untersuchungen (z. B. [SRS+07] und [RM08]) lassen sich Betrachtungsaspekte von Diensten ableiten, die z. B. zur Bewertung der Umsetzung dienstbasierter Architekturen in Unternehmen dienen (vgl. [RSE+07b]).

Unter *Wiederverwendbarkeit* wird die Eigenschaft eines Dienstes verstanden, ohne Anpassungen nur durch Parametrisierung in anderen als den ursprünglich geplan-

ten Einsatzszenarien verwendet werden zu können. Zur Sicherstellung der Wiederverwendbarkeit wird eine angemessene Dokumentation des Dienstes benötigt.

Das Konzept der *Granularität* beschreibt den Funktionsumfang sowie die Komplexität eines Dienstes. Dienste grober Granularität bilden ganze Funktionsbereiche innerhalb einer Domäne ab, z. B. einen bestimmten Geschäftsprozess. Dienste feiner Granularität stellen Basisfunktionalitäten zur Verfügung, die häufig in verschiedenen Szenarien verwendet werden können.

Die *Autonomie* ist ein Maß für die Unabhängigkeit eines Dienstes. Berücksichtigt wird hierbei die Abhängigkeit eines Dienstes von anderen Diensten oder Ressourcen. Ein Dienst ist dann autonom, wenn er unabhängig ist und die für sein Funktionieren notwendige Logik selbst bereitstellt. Er regelt alle seine Angelegenheiten selbstständig. Dies bezieht sich auch auf kompensierende Funktionen, um im Fehlerfall nicht auf eine übergeordnete Fehlerbehandlung angewiesen zu sein. Weiterhin bezieht sich die Autonomie eines Dienstes auch auf dessen funktionale Einzigartigkeit, d. h. dass die bereitgestellte Funktionalität nur vom betrachteten Dienst selbst angeboten wird.

Die *Kontextfreiheit* eines Dienstes liegt dann vor, wenn alle für die Nutzung notwendigen Daten im Dienstaufruf vorhanden sind und kein Zustand zwischen zwei Dienstaufrufen aufrecht erhalten werden muss. Es dürfen demnach keine Daten zwischen Dienstaufrufen vorgehalten werden.

Der *Kopplungsgrad* eines Dienstes ist ein weiteres Maß für die Unabhängigkeit eines Dienstes. Anzustrebendes Ziel beim Entwurf einer dienstbasierten Architektur ist hierbei die lose Kopplung, d. h. die vollkommene Unabhängigkeit und Austauschbarkeit eines Dienstes.[3]

Das *Geheimnisprinzip* im Sinne des Verbergens von Interna ist vor allem für Dienste, die von externen Dienstnutzern eingesetzt werden können, von hoher Wichtigkeit. Informationen, die nicht an externe Dienstnutzer weitergegeben werden sollen, umfassen unter anderem sicherheits- oder geschäftskritische Informationen sowie Implementierungsdetails.

Die *Auffindbarkeit* eines Dienstes stellt einerseits eine technische Anforderung dar. Dienste müssen vollständig und verständlich beschrieben und innerhalb eines Verwaltungssystems registriert sein um aufgefunden werden zu können. Die leichte Auffindbarkeit bereits vorhandener Dienste verhindert deren erneute Entwicklung („Parallelentwicklung“) und den unnötigen Zukauf von Diensten.

2.1.3 *Rollen in einer dienstbasierten Architektur*

Charakterisiert wird eine dienstbasierte Architektur ebenfalls durch das Zusammenspiel dreier Rollen, welche die Akteure innerhalb eines auf Diensten basierenden Systems annehmen können. Diese sind (vgl. z. B. [MES+08], [BFT04] und [Pap03]):

- Dienstanbieter
- Dienstnutzer
- Dienstvermittler

Als Dienstanbieter (engl. Service Provider) bezeichnet man in diesem Zusammenhang die Organisation bzw. Organisationseinheit, die einen Dienst zur Benutzung

3 Eine Diskussion der Kriterien für das Vorliegen loser Kopplung ist in [Loo03] zu finden.

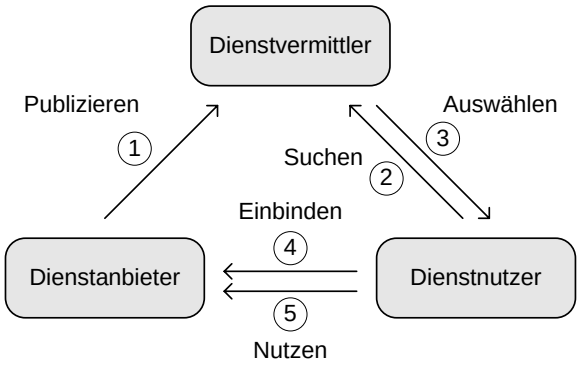


Abbildung 2: Rollen innerhalb einer dienstbasierten Architektur [MES+08]

bereitstellt. Ein Dienstanbieter kann sowohl unternehmensintern, z. B. eine Abteilung, als auch ein externer Dritter sein. Der Konsument eines Dienstes wird Dienstnutzer (häufig auch: Dienstnachfrager – engl. Service Requester) genannt. Optional sind auch Dienstvermittler (engl. Service Broker) denkbar, die zwischen Angebot und Nachfrage vermittelnd tätig werden. Ein Beispiel hierfür wäre ein Marktplatz, auf dem Dienste unterschiedlicher Anbieter gehandelt werden können.

Die Kommunikation zwischen den beteiligten Akteuren (Anbieter, Nutzer und Vermittler bzw. Intermediär) basiert auf dem Austausch von Nachrichten. Abbildung 2 stellt in Anlehnung an [MES+08] eine solche Kommunikation zwischen den drei Rollen dar. Der Anbieter stellt die Implementierung eines Dienstes zur Verfügung und veröffentlicht dessen Beschreibung sowie eine Schnittstellenbeschreibung in einem Verzeichnis beim Vermittler, das wiederum von den Kunden (Nutzern) durchsucht werden kann. Auf Grund der bei der Suche gefundenen Beschreibungen ist der Nutzer dann in der Lage, einen passenden Dienst auszuwählen, aufzurufen und somit zu nutzen. Anbieter und Nachfrager können auch ohne Interaktion mit einem Vermittler direkt zueinander in Kontakt treten.

Die eben beschriebene Aufteilung in drei Rollen, von denen die Rolle Dienstvermittler optional ist, wird in Weiterentwicklungen dienstbasierter Architekturen als nicht ausreichend bzw. zu unscharf betrachtet. In einem Internet der Dienste, dessen Vision eine Vielzahl frei handelbarer Dienste auszeichnet, die wie Ressourcen des heutigen Internets von jedermann erstellt und verwendet werden können, werden verschiedene ergänzende Rollen für komplexe dienstbasierte Systeme benötigt (vgl. z. B. [JRS08]). Die bekannten Rollen Dienstnutzer, Dienstanbieter sowie Vermittler von Diensten bleiben weiterhin bestehen, werden aber teilweise konkretisiert. Der Anbieter von Diensten muss nicht gleichzeitig der Produzent des Dienstes sein, so dass eine neue Rolle „Dienstproduzent“ mit den möglichen Unterrollen „Ingenieur“ (plant und konzipiert neue Dienste), „Entwickler“ (setzt diese Dienste in Programmcode um) und „Aggregator“ (aggregiert neue Dienstangebote aus bestehenden Diensten) definiert wird. Die Innovation neuer Dienste muss ebenfalls nicht originär vom Dienstanbieter oder Dienstproduzent übernommen werden, so dass hier eine neue Rolle „Dienstinnovator“ benötigt wird. Ebenfalls eine Konkretisierung erfährt die Rolle des Dienstvermittlers. Die ausschließliche Vermittlung von Diensten ist nicht die einzige Funktion, die eine dritte Partei innerhalb eines Szenarios wie dem des Internet der Dienste einnehmen kann. Ebenfalls unterhalb der Rolle „Drit-

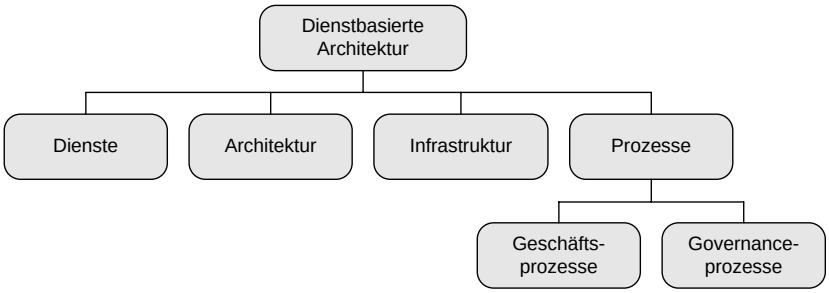


Abbildung 3: Bausteine aus fachlicher Sicht [Jos08]

te Partei“ als Spezialisierung zu verstehen ist die Funktion der Überwachung eines Dienstes zur Sicherstellung von Dienstgüteanforderungen der Dienstnutzer. In der vorliegenden Arbeit wird auf ein solch erweitertes Rollenverständnis zurückgegriffen, das neben den klassischen Rollen auch weitere dritte Parteien unterstützt.

2.1.4 *Bausteine dienstbasierter Systeme*

Neben den zuvor diskutierten Diensten und Rollen gibt es weitere Elemente, die bei der Realisierung eines dienstbasierten Systems berücksichtigt werden müssen.

Einer fachlich orientierten Betrachtung zufolge wird innerhalb eines dienstbasierten Systems der Dienst als Kernelement durch eine Architektur, eine geeignete Infrastruktur, Prozesse sowie Verfahren zur Governance unterstützt (siehe Abbildung 3 in Anlehnung an [Jos08]). Die Architektur setzt im Sinne des SOA-Paradigmas die Dienste zueinander in Verbindung. Die Infrastruktur ermöglicht die Nutzung von Diensten durch den Einsatz geeigneter Transportmechanismen, die Sicherstellung der technischen Interoperabilität zwischen den Diensten sowie die Überwachung von Dienstgüte- und Sicherheitsmerkmalen. Die Realisierung erfolgt in der Praxis durch einen Enterprise Service Bus (ESB), welcher eine Middleware für SOA darstellt. Prozesse besitzen in diesem Kontext aus zweierlei Sicht eine besondere Relevanz. Zum einen soll durch die Umsetzung eines Systems auf Basis von Diensten im Standardfall eine Reihe von Prozessen in flexibler Art und Weise abgebildet werden. Darüber hinaus werden auch im Rahmen der Realisierung sowie des Betriebs eines solchen dienstbasierten Systems Prozesse für das Management der Architektur und der Dienste benötigt. Governance beinhaltet alle Aspekte der Planung, Steuerung, Kontrolle eines solchen auf Diensten basierenden Systems. Hierzu zählen die strategische Ausrichtung der SOA, die Auswahl von Dienstanbietern sowie das Vertragsmanagement oder beispielsweise die Auswahl von Standards, die Einfluss auf das dienstbasierte System haben, und deren Umsetzung.

Mehr an der technologischen Umsetzung eines dienstbasierten Systems orientiert ist die Ausführung der notwendigen Elemente bei Krafzig et al. [KBS05]. Die Autoren zählen zu den Kernelementen eines dienstbasierten Systems, abgesehen von den Diensten selbst, die nachfolgenden Elemente (vgl. auch Abbildung 4): eine Benutzerschnittstelle, ein Metainformationssystem zur Diensteverwaltung (engl. Service Repository) sowie einen Service Bus, der mit der Infrastrukturkomponente bei Josuttis gleichzusetzen ist. Ebenfalls verfeinern die Autoren in diesem Kontext das Verständ-

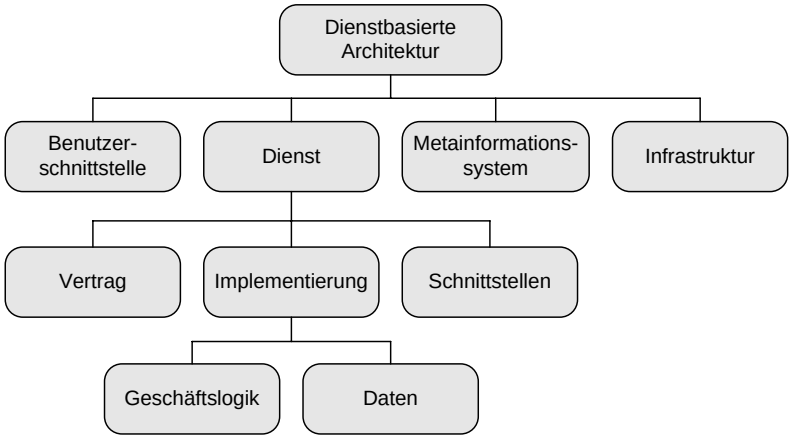


Abbildung 4: Bausteine aus technologischer Sicht [KBS05]

nis eines Dienstes. Ein Dienst kann nun abermals in einen Vertrag, Schnittstellen sowie eine Implementierung seiner Funktionalität unterteilt werden. Innerhalb eines Vertrags werden die funktionalen und nicht-funktionalen Merkmale des Dienstes innerhalb des Systems beschrieben (z. B. durch Vor- und Nachbedingungen), wohingegen die Schnittstellen die Funktionalität der Implementierung nach außen hin zur Verfügung stellen.

Für das weitere Verständnis dieser Arbeit sind sowohl technologische als auch fachliche Elemente von Relevanz. Neben den Diensten adressieren die Verfahren und Methoden, welche in dieser Arbeit entwickelt werden, in direkter Weise die Architektur, die zu Grunde liegende Infrastruktur sowie die involvierten Prozesse. Als Grundlage für die Verfahren und Methoden wird darüber hinaus das Vorhandensein eines Systems zur Verwaltung von Diensten vorausgesetzt.

2.2 GESCHÄFTSPROZESSE UND WORKFLOWS

2.2.1 *Definition und Grundlagen*

Ein Geschäftsprozess (im Folgenden auch als Prozess bezeichnet) ist definiert als Menge von Aktivitäten, die gemeinsam der Erreichung von Geschäftszielen dienen [Wes07]. Ein Geschäftsprozess besitzt einen eindeutigen Anfangszustand, einen Ablauf bestehend aus einer Reihe von Aktivitäten sowie einem definierten Endzustand. Hierbei wird gleichermaßen auf Mitarbeiter als auch auf technische Ressourcen zur Umsetzung der Aktivitäten zurückgegriffen. Unter einer Aktivität versteht man die kleinste ausführbare Einheit innerhalb eines Arbeitsablaufs.

Im Detail kann man einen Geschäftsprozess in der Realwelt im Sinne eines Prozessmodells verstehen, wobei eine Instanz eines solchen Modells, welche konkret ausgeführt wird, den eigentlichen Prozess darstellt [LR00].

Ein Workflow (Arbeitsablauf) ist der Bestandteil eines Geschäftsprozesses, der durch Informationstechnologie unterstützt bzw. ausgeführt wird. Ein Workflow muss nicht eine vollständige Automatisierung eines Prozesses darstellen, nutzt aber in Teilen technische Ressourcen zur Automatisierung [Wes07], [LR00], [JBS97], [AH02]. Da es sich bei einem Workflow um einen Ausschnitt eines Prozesses handelt, besteht

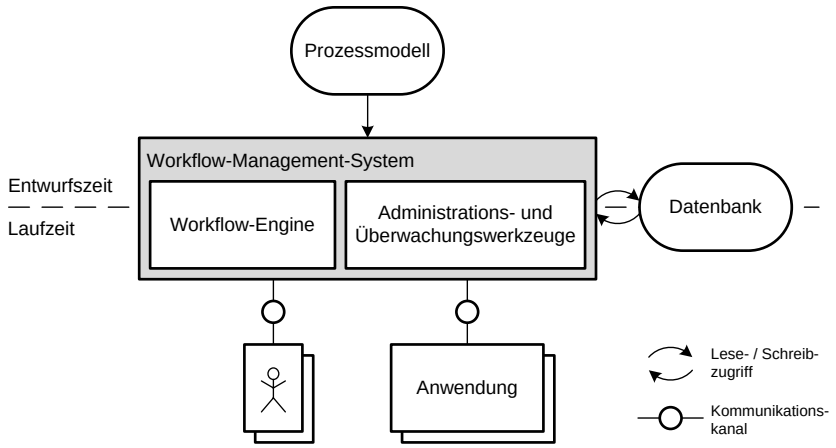


Abbildung 5: Kernkomponenten eines WfMS [Hol95]

auch ein Workflow wiederum aus einer Menge von Aktivitäten. In Analogie zu den Geschäftsprozessen lässt sich auch bei Workflows eine Unterscheidung in Modell und Instanz eines Modells vornehmen [LR00].

Unter dem Begriff Geschäftsprozessmanagement bzw. dem korrespondierenden Begriff Workflowmanagement werden alle Methoden und Konzepte zur Modellierung und Entwurf, der Umsetzung, der Ausführung und Steuerung von Geschäftsprozessen bzw. Workflows verstanden [Wes07].

Workflows lassen sich nach [LR00] anhand ihres Werts bzw. ihrer Bedeutung für ein Unternehmen sowie der Wiederholungsrate des Workflows in vier Klassen unterteilen: *Ad hoc*, *Kollaboration*, *Administration* und *Produktion*. In dieser Arbeit werden ausschließlich die sich durch eine hohe Wiederholungsrate und einen hohen Wertbeitrag auszeichnenden Produktionsworkflows betrachtet. Entsprechende Workflows implementieren die Kernprozesse eines Unternehmens. Auf Grund hoher Repetitionsraten besitzen sie ein hohes Automatisierungspotential und werden in Unternehmen bereits weitestgehend IT-unterstützt implementiert.

Stark strukturierte Workflow-Typen, zu denen nach der Klassifikation von [LR00] die Administrations- und Produktionsworkflows zählen, können durch so genannte Workflow-Management-Systeme (WfMS) unterstützt und ausgeführt werden. Abbildung 5 stellt die wichtigsten Elemente eines WfMS in Anlehnung an das WfMS-Referenzmodell der Workflow Management Coalition dar (vgl. [Hol95], [LR00] und [Mül05]). Die Modellierung in Abbildung 5 basiert auf einem Blockdigramm der *Fundamental Modeling Concepts* (FMC) Methode, welche für alle weiteren Architekturdarstellungen in dieser Arbeit verwendet wird (vgl. [KGT06]). WfMS sind Softwaresysteme, welche Workflows ausführen, steuern und überwachen. Sie bieten darüber hinaus Werkzeuge zur Erstellung von Prozessmodellen und Workflowdefinitionen, zur Simulation von Workflows sowie zu deren Verwaltung an. Ebenfalls sind sie für die Interaktion mit den die einzelnen Aktivitäten ausführenden Instanzen (bzw. Diensten – vgl. Abschnitt 2.2.2), wie z. B. Applikationen oder Personen, verantwortlich.

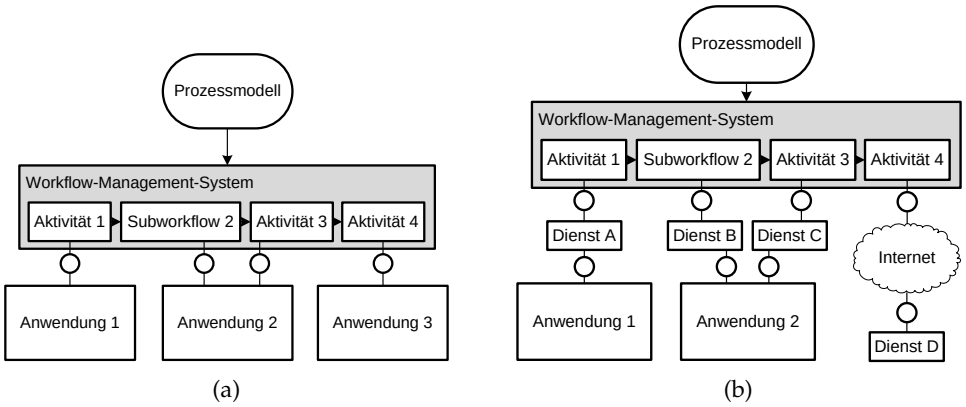


Abbildung 6: Vergleich traditioneller und dienstbasierter Workflows

Eine Implementierung einer Workflow-Engine im Kontext dienstbasierter Workflows auf Basis der Webservice-Technologie kann beispielsweise die in Abschnitt A.1.2 erläuterte BPEL-Engine sein.

2.2.2 *Dienstbasierte Workflows*

In diesem Abschnitt wird dargestellt, wie sich Workflows auf Basis einer dienstbasierten Architektur implementieren lassen. Grundlage für die so entstehenden dienstbasierten Workflows ist die Annahme, dass Bestandteile eines Workflows, also Subworkflows und Aktivitäten, durch einzelne Anwendungssysteme implementiert werden (vgl. hierzu auch Abbildung 6a). Anstelle des direkten Aufrufs eines solchen Anwendungssystems erfolgt beim dienstbasierten Workflow die Integration einer Indirektion, so dass an Stelle eines Anwendungssystems ein Dienst aufgerufen wird, der die Funktionalität des bestehenden Anwendungssystems kapselt oder eine vergleichbare Funktionalität anbietet (siehe Abbildung 6b). Dienste stellen somit eine Abstraktionsschicht zwischen einem Workflow-Management-System und den zugehörigen Anwendungen dar. Darüber hinaus sind sie Schnittstellen zu externen Dienstanbietern.

Eine Grundlage für die Umsetzung dienstbasierter Workflows ist das Finden und Zuordnen von Diensten zu den jeweiligen Bestandteilen von Workflows. So genannte Kompositionsverfahren bieten die Basis, um eine solche Selektion unter Berücksichtigung verschiedenster Auswahlkriterien automatisiert oder teilautomatisiert vornehmen zu können [BL06]. Beispiele für die Auswahl und Einbindung von Diensten unter der Berücksichtigung von Dienstgütemerkmalen finden sich z. B. bei Berbner et al. (vgl. [BGR+05], [BSR+06a], [BSR+06b]) oder Canfora et al. (vgl. [CPEV05], [CPEV08]) sowie in Abschnitt 2.3.2. Bevor eine solche Auswahl und Zuordnung allerdings vorgenommen werden kann, muss entschieden werden, welche Bestandteile eines Workflows durch welche Funktionen eines Dienstes abgebildet werden sollen. Hierbei spielt die Berücksichtigung der Granularität zur Verfügung stehender oder zu entwickelnder Dienste eine grundlegende Rolle. Beim so genannten (Zu-)Schnitt von Diensten können primär die beiden nachfolgenden Methoden unterschieden werden:

- *Top-Down*: Die Verknüpfung von Workflows mit Diensten erfolgt auf der Basis der Dekomposition, d. h. der Zerlegung von Workflow-Funktionen und Qualitätsanforderungen höherer Ebene auf solche tieferer Abstraktionsebenen (vgl. [Loco3]). Auf Basis einer vorhandenen Zerlegung in Subworkflows und Aktivitäten werden nachfolgend geeignete Dienste gesucht bzw. entwickelt, die die Funktion eines solchen Bestandteils implementieren.

- *Bottom-Up*: In diesem Absatz erfolgt die Definition von Diensten auf Basis der Analyse bestehender Anwendungslandschaften (vgl. Konzept der Service-Inventur [RSE+07b]). Es besteht die Möglichkeit, Funktionalitäten eines Anwendungssystems als Dienst zur Verfügung zu stellen, ohne das System selbst verändern zu müssen. Darüber hinaus kann ein Altsystem in Dienste zerlegt und neu zusammengesetzt werden, um dessen Wartbarkeit zu erhöhen.

Darüber hinaus sind verschiedene Mischformen der beiden Ansätze vorstellbar. Die Auswahl der Vorgehensweise hängt zumeist davon ab, welche Organisationseinheit innerhalb eines Unternehmens mit der Umsetzung dienstbasierter Workflows beauftragt wurde. Die Anwendung einer Top-Down-Vorgehensweise entspricht der einer Fachabteilung, wohingegen die Bottom-Up-Methode eher in einer IT-Abteilung zum Einsatz kommt. Für den weiteren Verlauf der Arbeit wird angenommen, dass eine Zuordnung zwischen Workflows und Kompositionen von Diensten existiert. Im Speziellen kommen hierbei Webservices zur Realisierung der Dienste zum Einsatz.

Ein Vorteil der Anwendung dienstbasierter Workflows besteht darin, dass durch das Beziehen einzelner Dienste von externen Anbietern Kostenreduktionen erreicht werden können, während die Steuerung des Workflows im Unternehmen verbleibt. Das Konzept gewährleistet ein hohes Maß an Flexibilität, da Dienste einfach ausgetauscht werden können, falls sie nicht die zugesicherten Leistungsmerkmale erfüllen. Eine Austauschbarkeit von Diensten reduziert auch die Abhängigkeit von externen Anbietern als auch von bestehenden Altsystemen. Dieser Ansatz ist demnach weitaus flexibler als traditionelles Business Process Outsourcing (BPO), das auf die Auslagerung eines vollständigen Geschäftsprozesses an externe Anbieter abzielt.

Der Einsatz dienstbasierter Workflows ermöglicht somit dienstbasiertes Outsourcing [BB03a], [BMS04], [SBM05]. Während das traditionelle BPO eine strategische Entscheidung mit Langzeitwirkung für ein Unternehmen war, ermöglichen dienstbasierte Workflows die Etablierung von operativen Outsourcing-Entscheidungen, die nur von kurzer Dauer sein können [BB03a].

2.3 DIENSTGÜTE

Dieser Abschnitt diskutiert das Konzept der Dienstgüte und dessen Bedeutung für dienstbasierte Systeme.

2.3.1 *Definition*

In der Literatur existiert für den Dienstgütebegriff keine allgemein akzeptierte Definition. Vielmehr sind die existierenden Definitionen stark vom Kontext ihrer jeweiligen Anwendung abhängig. Eine abstrakte Definition von Dienstgüte liefert die International Telecommunication Unit – Telecommunication Standardization Sector (ITU-T) in

ihrer Empfehlung „X.641 Information Technology - Quality of Service: Framework“, in welchem sie Dienstgüte wie folgt definiert [IT97]:

„A set of qualities related to the collective behaviour of one or more objects.“

Als Dienstgüte bezeichnet man demnach alle Qualitätsmerkmale eines Objekts gleich welcher Art. Ein weitere Konkretisierung des Dienstgüte-Begriffs findet sich bei [Sch01]:

„Dienstgüte kennzeichnet das definierte, kontrollierbare Verhalten eines Systems bezüglich quantitativ messbarer Parameter.“

Die beiden Definitionen stellen den Rahmen für die Definition dar, welche dieser Arbeit zu Grunde liegt. Während die Definition der ITU-T eine sehr große Generizität aufweist, schränkt die Definition von Schmitt das Verständnis von Dienstgüte auf rein quantitativ messbare Parameter ein. Im weiteren Verlauf der Arbeit wird hingegen ein Dienstgütebegriff zum Einsatz kommen, der sowohl qualitative als auch quantitative Aspekte berücksichtigt.

2.3.2 *Qualität dienstbasierter Systeme*

Die Definitionen von Dienstgüte des letzten Abschnitts haben ihren Ursprung im Umfeld der Kommunikationsnetze. Ebenfalls länger untersucht wird das Thema im Multimedia-Bereich, da z. B. Multimedia-Datenströme wie Sprach- und Videoübertragungen besondere Anforderungen in Bezug auf Dienstgüte besitzen [Ste00], [SN07], [MT04], [OSS02]. Die Behandlung von Dienstgüte für dienstbasierte Systeme und Workflows stellt dahingegen ein relativ neues Forschungsgebiet dar. Speziell die Integration externer Dienste im Sinne eines dienstbasierten Business Process Outsourcing erfordert ein dezidiertes Management der beteiligten Dienste. Ohne entsprechende Qualitätszusagen sind Anwender dienstbasierter Systeme nur in den seltensten Fällen dazu bereit, Dienste von Dritten einzukaufen [RM08], [TGNR03]. Darüber hinaus kommt der Dienstgüte-basierten Auswahl von Diensten eine wachsende Bedeutung zu [KKL03], [Ran03], [CPEV05], [CPEV08],[CSM+04].

Als Grundlage für ein umfassendes Dienstgütemanagement wird in einem nächsten Schritt ein Qualitätsmodell für dienstbasierte Systeme vorgestellt. Das in Abbildung 7 dargestellte Qualitätsmodell für dienstbasierte Systeme ist eine Anpassung sowie eine partielle Erweiterung des in [HSL+06] beschriebenen Qualitätsmodells für Peer-to-Peer-Systeme (P2P). Grundlegend werden in ihm die vier Qualitätsmerkmale *Adaptivität*, *Effizienz*, *Validität* und *Vertrauenswürdigkeit* unterschieden, welche sich wiederum in einzelne Untermerkmale untergliedern lassen.

In den folgenden Unterabschnitten werden (in Anlehnung an [HSL+06], [Ser05], [Men02] und [Ber08]) die wichtigsten der zur Sicherstellung hoher Dienstgüte bzw. Qualität dienstbasierter Systeme zu berücksichtigenden Merkmale und Untermerkmale behandelt. Der Fokus der Betrachtungen liegt im weiteren Verlauf auf der Anwendungsschicht als Teil der technischen Sicht sowie der Dienst- als auch der Workflowebene als Bestandteil der fachlichen Sicht. Für eine Diskussion von Dienst-

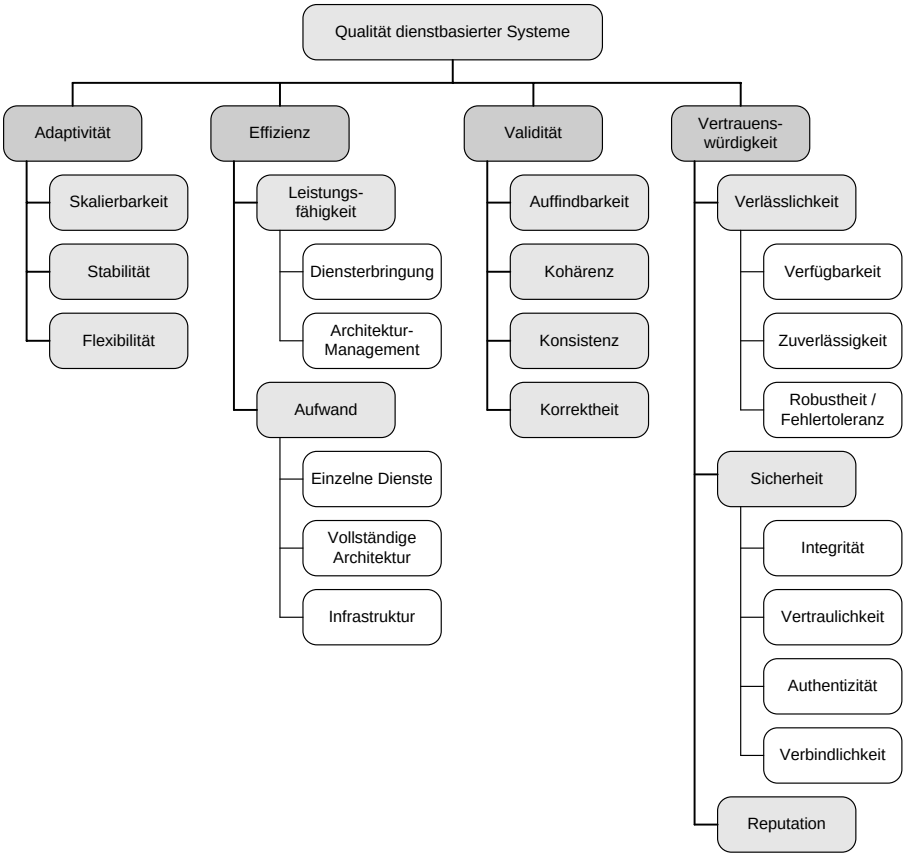


Abbildung 7: Qualität dienstbasierter Systeme (in Anlehnung an [HSL+06])

güteansätzen auf darunterliegender technischen Schichten, wie z. B. die *Integrated Services Architecture* (IntServ – vgl. [BCS94]) sowie die *Differentiated Services Architecture* (DiffServ – vgl. [BBC+98]), wird an dieser Stelle auf die einschlägige Literatur verwiesen, z. B. [Hec06], [XN99], [Sch01].

Adaptivität

Das Qualitätsmerkmal der Adaptivität kann in die drei Untermerkmale *Skalierbarkeit*, *Stabilität* sowie *Flexibilität* untergliedert werden. Hierbei wird unter der Skalierbarkeit eines dienstbasierten Systems dessen Anpassungsfähigkeit im Hinblick auf sich ändernde quantitative Anforderungen an das System verstanden, wie z. B. eine schwankende Nutzerzahl eines Dienstes oder der gesamten Infrastruktur.

Die Stabilität hingegen kennzeichnet die Fähigkeit eines dienstbasierten Systems bei geänderten Rahmenbedingungen weiterhin seine Funktionsfähigkeit aufrecht erhalten zu können. In einem dienstbasierten Szenario, in welchem Dienstkompositionen dynamisch auf Basis der Überwachungsergebnisse einzelner Dienstgütemerkmale durch Austausch von Diensten geändert werden können, kommt der Stabilität des Gesamtsystems eine besondere Rolle zu. Wechseln gleichzeitig verschiedene Dienstnutzer den Dienstanbieter auf Grund von Leistungsengpässen, so kann es innerhalb des Gesamtsystems zu Schwingungen kommen, falls es bei dem neu ausgewählten Dienstanbieter wiederum zu Überlastungserscheinungen kommt. Entsprechende

Phänomene lassen sich bei einer Vielzahl von komplexen Systemen beobachten (vgl. z. B. [Mog06]).

Die Flexibilität definiert die Anpassbarkeit an sich ändernde qualitative Anforderungen, d. h. den Einsatz des Systems oder einzelner Systemkomponenten innerhalb eines geänderten Umfelds. Die Flexibilität ist eines der grundsätzlichen Merkmale eines dienstbasierten Systems sowie eine der Grundanforderungen an einen Dienst. Ein Dienst soll auch in Anwendungsszenarien eingesetzt werden können, für welche er nicht konzipiert wurde. Das Merkmal ist eng verknüpft mit den Konzepten der Wiederverwendbarkeit, der Autonomie sowie der Kontextfreiheit, wie sie in den Betrachtungsaspekten von Diensten in Abschnitt 2.1.2 aufgeführt werden.

Effizienz

Die Effizienz eines dienstbasierten Systems wird definiert als das Verhältnis seiner Leistungsfähigkeit zu dem Aufwand, welcher zur Realisierung der Leistungsfähigkeit benötigt wird [HSL+06].

Bei Betrachtung der Leistungsfähigkeit (auch: Performanz) eines dienstbasierten Systems können die Leistungsfähigkeit der Diensterbringung sowie die des Architekturmanagements unterschieden werden. Im Sinne dieser Arbeit wird die Leistungsfähigkeit der Diensterbringung (in Anlehnung an [BHM+04], [LJL+03], [SL05]) durch die Antwortzeit und den Durchsatz eines Dienstes definiert. Die Antwortzeit beinhaltet die Verarbeitungszeit in Start- und Endknoten, d. h. der Generierung einer Anfrage beim Dienstnutzer bzw. der Bearbeitung beim Dienstanbieter, die Transportzeit (Latenz) auf dem Netz sowie der Verarbeitungszeit in den Netzkomponenten. Am Beispiel der Webservice-Technologie gehen zusätzlich zu den Verarbeitungszeiten in den Protokollstapeln von Start- und Endknoten sowie der Transportzeit auch Zeiten für die Serialisierung von Dienstaufrufen in XML und zurück sowie einer möglichen Kompression und Verschlüsselung der Nachrichten mit in die Antwortzeit ein. Weitere Bestandteile sind die Zeit für den Verbindungsauf- und -abbau, für die Verhandlung von Verbindungsparametern sowie für die Authentifizierung. Im Fehlerfall wird die Antwortzeit als die Zeit definiert, die von der Anfrage bis zum Empfangen einer Fehlermeldung bzw. Feststellen eines Fehlers vergeht.

Der Durchsatz eines Dienstes beschreibt die Fähigkeit eines Webservices zur Verarbeitung gleichzeitiger Anfragen und wird gemessen in Verbindungen, Dienstinteraktionen (bestehend aus einem Aufruf und zugehöriger Antwort) oder auch Datenpaketen pro Zeiteinheit. Je nach Protokollschicht können unterschiedliche Verbindungstypen Gegenstand der Betrachtung sein. Beim Einsatz von Webservices können dies z. B. TCP- (Transmission Control Protocol) und HTTP-Verbindungen oder aber SOAP-Interaktionen sein.

Es existieren diverse Ansätze für das Performanz-Management von Diensten auf Webservice-Basis. Eckert et al. verwenden Network Calculus und Warteschlangentheorie, um die Leistungsfähigkeit einzelner Dienste sowie kompletter Workflows abzuschätzen [EPR+07], [ESN+08]. Die Analyse von Webservice-Performanz zur Steigerung der Zuverlässigkeit dienstbasierter Workflows über verschiedene Protokollschichten hinweg ist Gegenstand der Arbeiten von [RBHS07], [RBE+07].

In Ergänzung zu der Betrachtung einzelner Dienste lässt sich auch die Leistungsfähigkeit des gesamten Architekturmanagements feststellen. Zum Architekturmanagement zählen alle Operationen, die zur Aufrechterhaltung der Funktionsfähigkeit und

Qualität des dienstbasierten Systems bzw. Workflows und der Infrastruktur verwendet werden. Hierzu zählen Dienstauswahl- und Kompositionsverfahren ebenso wie Verfahren zur Überwachung von Dienstgütemerkmalen innerhalb der Infrastruktur.

Zur Realisierung des gewünschten Leistungsniveaus eines dienstbasierten Systems müssen Ressourcen eingesetzt bzw. genutzt werden. Ein solcher Ressourceneinsatz wird als Aufwand definiert. Nach [HSL+06] muss der Aufwand für einzelne Dienste, für die vollständige Architektur sowie die zu Grunde liegende Infrastruktur gesondert betrachtet werden. Ressourcen innerhalb eines dienstbasierten Systems können beispielsweise die Dimensionierung der Netzwerkkapazitäten darstellen, über welche Dienste genutzt werden oder aber die Ausstattung der die Dienste anbietenden Applikationsserver. Da sich die Ermittlung des Ressourcenbedarfs der unterschiedlichen Komponenten aus fachlicher Sicht zumeist nicht realisieren lässt, werden die in Dienstgütevereinbarungen definierten Kosten als Kenngröße für den Aufwand verwendet.

Validität

Das Qualitätsmerkmal der Validität bezieht sich im Kontext dienstbasierter Systeme auf das Management des Lebenszyklus eines Dienstes bzw. auf die Verwaltung aller in einem System genutzten Dienste. Das Merkmal lässt sich in die Untermerkmale *Auffindbarkeit*, *Kohärenz*, *Konsistenz* und *Korrektheit* untergliedern.

Die einfache Auffindbarkeit von Diensten ist eine Notwendigkeit für eine effiziente Wiederverwendung (vgl. Abschnitt 2.1.2). Nur eine angemessene Annotation von Diensten, z. B. auf Basis semantischer Technologien (vgl. [SERS08]), und der Einsatz geeigneter Auswahlverfahren zum Finden der in der jeweiligen Situation benötigten Dienste, erlauben einen hohen Wiederverwendungsgrad sowie die Verhinderung von Neuentwicklungen bereits bestehender Dienste.

Sowohl das Merkmal der Kohärenz als auch das der Konsistenz adressieren die Aktualität eines oder mehrerer Dienste innerhalb eines dienstbasierten Systems. Die Kohärenzanforderung drückt dabei aus, dass gleichzeitig unterschiedliche Versionen eines Dienstes im Einsatz sein können. Ursache hierfür kann beispielsweise die unterschiedliche Aktualität der Verzeichnisdienste sein, über welche die Dienste gesucht werden, so dass parallel verschiedene Versionen eines Dienstes angeboten werden. Hieraus resultiert, dass ein Dienstanbieter nicht einfach alte Versionen außer Betrieb nehmen kann, sondern diese vielmehr vorhalten muss. Es kann sich um eine bewusste Entscheidung eines Dienstnutzers handeln, nicht die neuste Version eines Dienstes einzusetzen. Die Kohärenzanforderung lässt sich durch Gültigkeitsintervalle für Versionen steuern, nach deren Ablauf die Dienstnutzer nach einer aktuellen Version des Dienstes suchen müssen. Die Konsistenz hingegen verschärft das Kohärenzmerkmal nochmals. Die Konsistenz fordert, dass alle Dienstnutzer jeweils auf einer einheitlichen bzw. der aktuellsten Version eines Dienstes arbeiten sollten. Eine solche Forderung lässt sich aber nur im unternehmensinternen Einsatz dienstbasierter Systeme realisieren. Sowohl Kohärenz als auch Konsistenz stellen Anforderungen dar, die für das Dienstmanagement zur Komplexitätsreduktion wünschenswert sind. Allerdings kann von beiden Anforderungen jederzeit aus praktischen Gründen abgewichen werden.

Das Untermerkmal der Korrektheit repräsentiert die funktionalen Anforderungen an einen Dienst oder an ein dienstbasiertes System. Die Korrektheitsanforderung wird durch Vor- und Nachbedingungen ausgedruckt, welche durch Testverfahren auf ihre Einhaltung hin überprüft werden können. Ja nach Betrachtungsgegenstand kommen hierbei Komponenten-, Integrations- oder auch Systemtestverfahren zum Einsatz.

Vertrauenswürdigkeit

Das Merkmal der Vertrauenswürdigkeit eines dienstbasierten Systems kann in die Untermerkmale *Verlässlichkeit*, *Sicherheit* und *Reputation* unterteilt werden. Der Vertrauenswürdigkeit eines dienstbasierten Systems kommt gerade im Kontext unternehmensübergreifender Dienstnutzung ein hoher Stellenwert zu, da Geschäftsprozesse in Teilen nach Außen hin geöffnet werden.

Die Verlässlichkeit von Diensten und dienstbasierten Systemen kann ihrerseits in die Untermerkmale *Verfügbarkeit*, *Zuverlässigkeit* und *Robustheit* (auch: Fehlertoleranz) unterschieden werden.

Die Verfügbarkeit eines Dienstes oder eines gesamten Systems charakterisiert dessen Fähigkeit, zu einem bestimmten Zeitpunkt die von ihm geforderte Leistung erbringen zu können. Im Kontext dienstbasierter Workflows stellt das Merkmal Verfügbarkeit eines der wichtigsten Dienstgütemerkmale dar. Ein nicht verfügbarer Dienst als Teil einer Komposition von Diensten verhindert im schlimmsten Fall die Ausführung des vollständigen Workflows [PSL03], [Men02], [LJL+03]. Die Verfügbarkeit ist definiert als Wahrscheinlichkeit, mit der ein Dienst zu einem bestimmten Zeitpunkt genutzt werden kann. Sie kann als Verhältnis der mittleren Zeit bis zum Auftreten eines Fehlers, welche auch als mittlere Lebensdauer bezeichnet wird, zur Summe aus mittlerer Lebensdauer und mittlerer Fehlerbehebungszeit ausgedrückt werden [Ser05]. Zur Verbesserung der Verfügbarkeit kommen Replikationsstrategien in Verbindung mit Verfahren zur Kapazitätsplanung (vgl. [ERS+07], [ESR+08], [EEM+08]) sowie Ansätze zur Beschleunigung der Fehlerkorrektur zum Einsatz. Ebenfalls von zunehmender Bedeutung sind in diesem Zusammenhang Sicherheitsaspekte, beispielsweise zur Verhinderung von „Denial of Service“ Attacken, deren Ziel die Einschränkung der Verfügbarkeit eines Dienstes ist (siehe z. B. [Eck07] und [Tan02]).

Die Zuverlässigkeit eines Dienstes beschreibt die Wahrscheinlichkeit, mit welcher der Dienst zu einem bestimmten Zeitpunkt die geforderte Funktionalität zur Verfügung stellen kann [Som04], [Ser05]. Anders ausgedrückt wird hiermit beschrieben, in wie weit der Dienst seiner Spezifikation entspricht und aus Sicht des Dienstnutzers das tut, was er tun soll. Mangelnde Zuverlässigkeit wird einem Dienst häufig dann bescheinigt, wenn bei der Dienstnutzung Fehler auftreten. Dies kann sich durch fehlerhafte Ergebnisse eines Dienstes ebenso äußern, wie beispielsweise durch verzögerte oder verlorene Nachrichten oder durch Nichterreichbarkeit eines Dienstes. Die Zuverlässigkeit ist demnach kein ausschließlich objektiv messbares Merkmal. Vielmehr spielt die subjektive Komponente eine starke Rolle – das Erfüllen der erwarteten Anforderungen an einen Dienst kann in Abhängigkeit des Dienstnutzers sehr unterschiedlich sein. Mögliche Kennziffern zur Bestimmung der Zuverlässigkeit eines Dienstes oder dienstbasierten Systems sind die Fehlerhäufigkeit als Anzahl der fehlerhaften Dienstaufrufe im Verhältnis zur Anzahl aller Dienstaufrufe, die Fehlerwahrscheinlichkeit sowie die mittlere Betriebsdauer zwischen Ausfällen als Erwar-

tungswert für den zeitlichen Abstand zwischen dem Auftreten zweier Fehler beim selben Dienst. Ansätze, die die Zuverlässigkeit eines Dienstes oder dienstbasierten Systems verbessern, sind z. B. Transaktionskonzepte und Kompensationsmechanismen (siehe z. B. WSBPEL in Abschnitt A.1.2). Diese sorgen im Falle des Auftretens eines Fehlers für die Wiederherstellung eines ordnungsgemäßen Zustands, z. B. im Sinne des ACID-Konzepts (Atomicity, Consistency, Isolation und Durability – vgl. [Bir05]) für Transaktionen.

Als Robustheit wird die Fähigkeit eines Dienstes oder dienstbasierten Systems verstanden, den Systembetrieb unter der vorherrschenden Dynamik des Systems und bei Störungen aufrecht erhalten zu können. Unter der Dynamik eines dienstbasierten Systems wird an dieser Stelle das Austauschen von funktionsgleichen Diensten zur Laufzeit verstanden, wie sie im Sinne der losen Kopplung möglich ist. Zur Gewährleistung der Robustheit kommen fehlertolerante Systeme (z. B. [AHA+00], [BM99], [BCCT05]), Verfahren zur Selbstheilung und Selbstmanagement von Systemen (z. B. [DPTS06], [MLH+06], [MMP06]) bzw. Prinzipien des Autonomic Computings (z. B. [HM08]) zum Einsatz. Gemeinsames Ziel dieser Ansätze ist die automatisierte Wiederherstellung eines stabilen Systemzustands.

Die Sicherheit eines dienstbasierten Workflows ist von hoher Relevanz für die Akzeptanz dienstbasierter Systeme im Unternehmenskontext, da die Öffnung von Geschäftsprozessen bei der Einbindung externer Dienstanbieter ein Unternehmen potentiell verwundbar macht. Sicherheitsanforderungen für dienstbasierte Systeme lassen sich einerseits aus den Anforderungen „klassischer" verteilter Systeme ableiten (vgl. z. B. [Eck07]) und sollten bereits bei der Entwicklung eines solchen Systems methodisch unterstützt Berücksichtigung finden (z. B. durch Einsatz geeigneter Entwurfsmuster – vgl. [SR01] und [SFBH+05]). Andererseits induziert die Verwendung dienstbasierter Architekturen eine Reihe zusätzlicher Herausforderungen [HB09]. Hierbei gilt es insbesondere die lose Kopplung von Diensten an Workflows, die Aufteilung (Dezentralität) der Prozesskontrolle auf Dienstnutzer und -nachfrager sowie die Heterogenität der die Dienste bereitstellenden Systeme zu nennen. Auch das Merkmal der Sicherheit lässt sich wiederum in mehrere Untermerkmale unterscheiden (in Anlehnung an [Bun08] und [HSL+06]): *Integrität*, *Vertraulichkeit*, *Authentizität* und *Verbindlichkeit*.

Die Integrität beschreibt die Eigenschaft eines dienstbasierten Systems für jede ausgetauschte Nachricht zu gewährleisten, dass eine unbemerkte und unautorisierte Veränderung einer Nachricht ausgeschlossen werden kann. Da Nachrichten nacheinander von verschiedenen Instanzen verarbeitet werden, die einzelne Elemente einer Nachricht verändern, muss eine partielle Sicherung einer Nachricht auf Elementebene die vollständige Sicherung einer Nachricht ersetzen.

Die Vertraulichkeit einer Nachricht, d. h. die Geheimhaltung ihrer Inhalte, wird durch Verschlüsselungsverfahren auf Nachrichtenebene realisiert. Auch hier gilt, dass Nachrichten von verschiedenen Instanzen verarbeitet werden, diese aber nicht die Nachricht vollständig lesen sollen. Die partielle Verschlüsselung der Nachricht kann hier ein Ausweg sein, wobei die Schlüsselverteilung innerhalb eines dynamischen und heterogenen Systems besonders berücksichtigt werden muss. Eng hiermit verknüpft ist die Autorisierung des Zugriffs auf die Inhalte einer Nachricht bzw. der Berechtigung zur Nutzung eines Dienstes. Im klassischen Unternehmensumfeld wird dies in Form zentral verwalteter Berechtigungsprofile abgebildet, welche sich aber

innerhalb eines dezentral kontrollierten dienstbasierten Systems nur eingeschränkt anwenden lassen. Als Lösung kann die Autorisierung auf Nachrichtenbasis umgesetzt werden.

Das Untermerkmal der Authentizität charakterisiert die Möglichkeit zur Identifizierung eines Dienstnutzers gegenüber einem Dienst ebenso wie der Bestätigung der Identität eines Dienstanbieters gegenüber dem Nutzer des Dienstes. Herausforderung in dienstbasierten Systemen ist die Heterogenität der Authentifizierungsmechanismen, die bei unterschiedlichen und wechselnden Dienstanbietern zum Einsatz kommen. Die Authentizität besitzt gerade im Hinblick auf die Abrechnung von Diensten eine große Wichtigkeit.

Die Verbindlichkeit einer Dienstnutzung bzw. -angebots ist von hoher Relevanz, da durch Dienstaufrufe geschäftliche Transaktionen abgebildet werden, die ihrerseits Rechtsfolgen bedingen. Es muss jederzeit sicher nachgewiesen werden können, wer an welcher Transaktion beteiligt war. Somit wird verhindert, dass eine Dienstnutzung nachträglich abgestritten werden kann.

Darüber hinaus spielen zunehmend die Gewährleistung der Einhaltung von internen und externen Rahmenbedingungen, wie z. B. die Prüfbarkeit dienstbasierter Systeme aus rechtlicher Sicht oder die Berücksichtigung von Datenschutzaspekten (vgl. [Tur09]), sowie die Verwaltung verteilter Sicherheitslösungen eine grundlegende Rolle, auf deren weitere Diskussion ebenso wie die Darstellung möglicher Gefährdungspotentiale an dieser Stelle verzichtet werden soll. Es wird hier auf [Eck07] und [Bun08] verwiesen.

Ansätze und Spezifikationen für die Sicherung von Webservices ergänzen bisherige Schutzmaßnahmen für internetbasierte Systeme, wie z. B. SSL (Secure Sockets Layer) oder TLS (Transport Layer Security) unter Verwendung symmetrischer und asymmetrischer Kryptografie-Verfahren (vgl. z. B. [Buc08]). Beispielsweise kann zur Authentifizierung von Webservice-Aufrufen die von OASIS spezifizierte Security Assertion Markup Language (SAML) verwendet werden (z. B. [MES+08], [Jos08], [KC08]), wohingegen die Verschlüsselung von Nachrichtenteilen sowie der Schlüsseltausch durch den WS-Security-Standard abgebildet werden können (vgl. [ACKM04], [MES+08], [RR04]). Die gerade in dezentralen Szenarien notwendige Gewährleistung der Vertrauenswürdigkeit, z. B. bei der Abrechnung erbrachter Leistungen, kann durch ein Token-basiertes Verfahren realisiert werden (vgl. [Lie08] für eine Anwendung im P2P-Umfeld).

Weiteres Untermerkmal der Vertrauenswürdigkeit ist die *Reputation* eines Dienstes oder dienstbasierten Systems. Sie beinhaltet Erfahrungen mit einem Dienst bzw. dienstbasierten System sowie das Vertrauen einer Anzahl von Nutzern in dessen Funktionsweise und Leistungsfähigkeit. Reputation stellt gleichermaßen die Erwartung an ein zukünftiges Verhalten eines Dienstes oder Systems dar [BF08]. Sie wird zumeist auf Basis eines Ratings, d. h. der Bewertung verschiedener dienstspezifischer Kriterien, ermittelt. Eine solche Bewertung gibt die Erfahrungen eines Nutzers wieder. Eine hohe Reputation hat einen positiven Wert für einen Dienstanbieter. Sie erhöht die Wahrscheinlichkeit, dass ein Dienst von einem potentiellen Nutzer ausgewählt wird. Dienstanbieter können beispielsweise Referenzen auf bestehende Geschäftsbeziehungen nutzen, um ihre Reputation zu steigern.

Reputationsmechanismen im Kontext dienstbasierter Workflows spielen im Besonderen im Bereich der automatisierten Komposition von Diensten zu Workflows eine

große Rolle. So beschreibt Maximilien et al. ein agentenbasiertes System zum Finden und Auswählen von Diensten anhand der Reputation des Dienstanbieters [MS02b], [MS02a]. Berbner et al. setzen ebenfalls die Reputation eines Dienstanbieters als Selektionskriterium für Dienste ein [BGR+05]. Reputation als Bestandteil der automatisierten Verhandlung von Leistungsvereinbarungen ist Gegenstand der Arbeit von [KHEP08].

2.3.3 *Dienstgütevereinbarungen*

In diesem Abschnitt wird das Konzept der Dienstgütevereinbarung eingeführt und erläutert. Die Diskussion beschränkt sich hierbei auf die technischen Aspekte einer solchen Vereinbarung – rechtliche Aspekte werden nicht behandelt.

Eine Dienstgütevereinbarung stellt einen schriftlichen Vertrag zwischen einem Anbieter und einem Nutzer eines Dienstes dar, in welchem verschiedene Leistungsaspekte einer Diensterbringung vertraglich festgehalten werden [SDM01], [ULMS08], [MMBBD02]. Entsprechende Verträge können zwischen Anbietern und Nutzern aus verschiedenen Unternehmen definiert werden, aber auch zwischen Parteien innerhalb eines Unternehmens, wie z. B. verschiedenen Abteilungen [EK04]. Durch eine Dienstgütevereinbarung verpflichtet sich ein Dienstanbieter zur Erbringung eines vordefinierten Leistungsniveaus zu einem vordefinierten Zeitpunkt. Ebenfalls kann innerhalb der Vereinbarung definiert werden, welche Vertragsstrafen im Falle der Verletzung der Vereinbarung fällig werden. Grundlegende Bestandteile, welche innerhalb einer Dienstgütevereinbarung festgelegt werden, sind [ULMS08], [LKD+03a]:

- Die Definition der involvierten *Vertragspartner* (engl. Parties).
- Die Festlegung der zu betrachtenden *Dienstgüteparameter* (engl. Service Level Parameters), welche nachprüfbar bzw. messbar sein müssen. Hierbei ist es wichtig, dass beide Vertragspartner ein gemeinsames Verständnis von einzelnen Parametern und den zugehörigen Metriken haben.
- Die Bestimmung der *Ziele* (engl. Service Level Objective – SLO) in Bezug auf die für verschiedene Parameter zu erbringenden Leistungen.

Im Umfeld dienstbasierter Systeme existieren zahlreiche Konzepte und Spezifikationen für die Beschreibung von Dienstgütemerkmalen und -vereinbarungen. Nachfolgend werden ausgewählte Beispiele vorgestellt, die sowohl generisch anwendbar als auch speziell auf eine Anwendung im Webservice-Umfeld zugeschnitten sind.

Ein Ansatz, der die unternehmensübergreifende Verhandlung und Spezifikation von Verträgen ermöglicht, ist die Business Contract Language (BCL – vgl. [NCL+03] und [LMC+04]. Die BCL ist eine XML-basierte Sprache, die es erlaubt, verschiedene Typen von verhaltensrelevanten Regelungen, wie Pflichten, Erlaubnisse oder Verbote, innerhalb einer unternehmensübergreifenden Geschäftsbeziehung zu beschreiben. Auch wenn BCL ein domänenunabhängiger Ansatz zur Beschreibung von Geschäftsbeziehungen ist, lassen sich Dienstgütevereinbarungen durch die BCL ausdrücken.

Ein Ansatz, der speziell auf die Anforderungen von Webservices zugeschnitten ist, ist der durch IBM entwickelte Web Service Level Agreements (WSLA) Ansatz [LKD+03a], [LKD+03b], [LDK04], [KL03], [DDK+04]. WSLA beinhaltet neben einer umfangreichen XML-basierten Beschreibungssprache für Dienstgütevereinbarun-

gen ein Templating-System für Vereinbarungen sowie ein Dienstgütemanagement-Framework für Webservices. Zielsetzung des WSLA-Ansatzes ist die Unterstützung des vollständigen Prozesses von Erstellung, d. h. Verhandlung und Abschluss, Umsetzung und Überwachung von Dienstgütevereinbarungen in Webservice-Szenarien. Die Struktur einer WSLA-konformen Dienstgütevereinbarung bildet die oben aufgeführten Konzepte der Vertragspartner und deren Rollen, der Dienstgüteparameter und den zugehörigen Metriken, sowie die Definition von Zielen ab. Es wird definiert, welche der Parteien für die Durchführung von Messungen verantwortlich ist, und wie im Falle von Abweichungen von Zusicherungen verfahren wird. WSLA besitzt eine hohe Durchdringung im Forschungsumfeld, da es zu einem frühen Zeitpunkt einen großen Umfang an Funktionsmerkmalen zur Verfügung stellte.

Einen Ansatz zum automatisierten Verhandeln von Dienstgütevereinbarungen im Grid-Umfeld stellt das durch das Open Grid Forum standardisierte WS-Agreement dar [ACD+07].[4] Die Zielsetzung des WS-Agreement-Standards ist die Spezifikation einer Sprache und des zugehörigen Interaktionsprotokolls, mit deren Hilfe die aktuellen Fähigkeiten und Kapazitäten eines Grid-Dienstanbieters einer Anzahl von Interessenten kommuniziert werden und auf deren Basis Verhandlungen über die angebotenen Leistungsmerkmale geführt werden können. Ziel ist auch hier die Automatisierung von Verhandlungen in dynamischen Umfeldern sowie die Unterstützung einer möglichen Überwachung von zugesicherten Leistungsmerkmalen auf Basis einer maschinenlesbaren Anforderungssprache.

Einen weiteren Beschreibungsrahmen für die Anforderungsdefinition auf XML-Basis stellt das durch das W3C spezifizierte WS-Policy-Framework zur Verfügung [BBC+06]. WS-Policy definiert hierbei ein Rahmenwerk, welches durch Instanziierung der WS-Policy-Sprache und Konkretisierung für spezielle Anwendungsfelder angepasst werden muss. Das Framework erlaubt die Beschreibung von Fähigkeiten, Anforderungen, Zusicherungen sowie anderen Merkmalen eines Webservice-basierten Systems.

Abschließend gilt es zu erwähnen, dass wichtige Merkmale der Gestaltung von Dienstgütevereinbarungen für Webservices die Berücksichtigung von Operationen und deren Bindungen an Protokolle und Datenformate sind. Verschiedene durch einen Webservice angebotene Protokolle und Nachrichtenformate bedingen unterschiedliche Dienstgütevereinbarungen.

2.3.4 *Überwachung von Dienstgütemerkmalen*

Grundlagen des Dienstgütemanagements

Im Umfeld dienstbasierter Workflows gilt es nicht nur die Dienstgüte einzelner Dienste zu spezifizieren und nachfolgend im Betrieb zu überwachen – vielmehr müssen Kompositionen von Diensten aus Dienstgütesicht betrachtet werden. Dies bedeutet im Detail, dass, basierend auf Dienstgüteanforderungen an den gesamten Workflow, die Anforderungen an einzelne Dienste ermittelt werden müssen. Während der so genannten dienstgütebezogenen Komposition (engl. QoS-aware Service Composition) erfolgt durch algorithmische Unterstützung die Selektion geeigneter Dienste auf Basis der durch die Dienste bereitgestellten Dienstgütemerkmale (vgl.

4 Der Vorläufer von WS-Agreement ist WS-Negotiation, eine Sprache und Protokoll, welches sich immer noch in zahlreichen Projekten im Einsatz befindet [HLJ04].

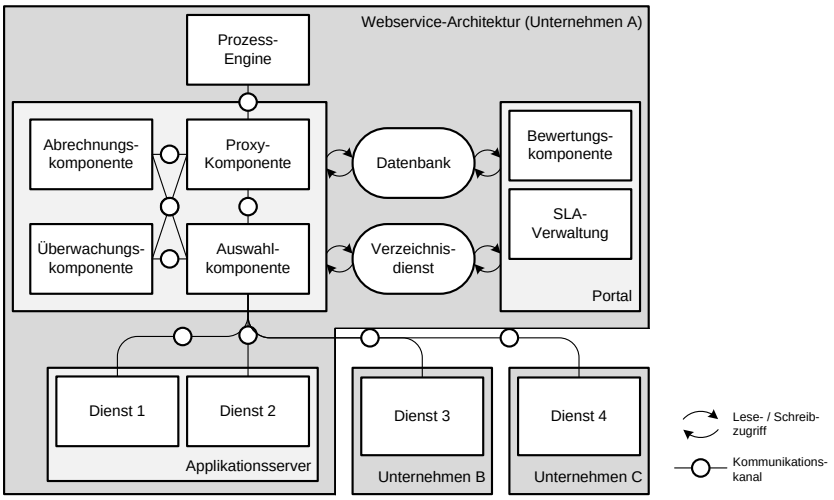


Abbildung 8: Beispiel eines Dienstgütemanagementsystems [BHMS05]

z. B. [JMS02b], [CSM+04], [KKL03], [Ran03]). Geeignete Dienste werden aus entsprechenden Dienstkatalogen entnommen. Zur Laufzeit des dienstbasierten Workflows muss dessen Dienstgüte laufend überwacht werden. Eine solche Überwachung erfolgt wiederum sowohl auf Ebene des Workflows, als auch auf Ebene der einzelnen Dienste. Sollten zur Laufzeit Verletzungen der an den Workflow oder an einzelne Dienste gestellten Anforderungen festgestellt oder erwartet werden, so gilt es adäquate Gegenmaßnahmen, wie z. B. die Neuauswahl eines alternativen Dienstes ([CPEV05], [CPEV08]), zu starten.

Zur technischen Abbildung dieser Anforderungen kommen Dienstgütemanagementsysteme zum Einsatz, die den vollständigen Lebenszyklus einzelner Dienste sowie dienstbasierter Workflows abbilden. Ein Vertreter hierfür ist das in Abbildung 8 dargestellte WSQoSX-Framework[5] (Web Service Quality of Service Architectural Extensions). Ein auf dem WSQoSX-Framework basierendes System durchläuft beim Management von Dienstgüte die folgenden Phasen [BHMS05]:

1. Ermittlung der Anforderungen an den Gesamtworkflow sowie an die zugeordneten abstrakten Dienste (Komponenten *Prozess-Engine*, *SLA Verwaltung*).[6]

2. Modellierung bzw. Transformation der Anforderungen in maschinenlesbare Dienstgütevereinbarungen (Komponenten *SLA Verwaltung*, *Datenbank*).

3. Umsetzung des dienstbasierten Workflows durch Implementierung bzw. Komposition von konkreten Diensten (Komponenten *Proxy*, *Auswahl*, *Verzeichnisdienst*).

4. Überwachung der Dienstgüte zur Ausführungszeit des Workflows durch geeignete Instrumentierungsverfahren (Komponenten *Prozess-Engine*, *Proxy*, *Auswahl*, *Überwachung*, *Datenbank*, optional *Abrechnung*).

5 Für eine detaillierte Beschreibung wird an dieser Stelle auf [BHMS05], [BGR+05], [BGR+07] und [Ber08] verwiesen.

6 Die Unterscheidung in abstrakte und konkrete Dienste ermöglicht die Modellierung von „Platzhaltern" innerhalb des Workflows, welche später durch geeignete Auswahl existierender Dienste ersetzt werden können.

5. Vergleich der Messwerte mit den in den Dienstgütevereinbarungen modellierten Anforderungen (Komponenten *Überwachung*, *Datenbank*, *SLA Verwaltung*).

6. Reporting von Abweichungen von Soll- zu Ist-Zustand und Reaktion auf diese Abweichungen (Komponenten *Prozess-Engine*, *Proxy*, *Bewertung*, *Auswahl*).

Hierbei kann man die Phasen 1-3 der Entwurfs- und Entwicklungsphase eines dienstbasierten Workflows zuordnen, wohingegen die Phasen 4-6 der Ausführung eines Workflows zuzuordnen sind. Rückkopplungen zwischen Entwurfs- und Entwicklungsphase sowie der Ausführung sind möglich, z. B. im Falle des erneuten Auswählens eines Ersatzdienstes oder bei der Neuverhandlung von Dienstgütevereinbarungen.

Die Betrachtung der Ausführungsphase ist Schwerpunkt der weiteren Arbeit. Insbesondere die Überwachung von Diensten sowie die Reaktion auf Abweichungen von vereinbarten Dienstgütemerkmalen sind Gegenstand dieser Arbeit.

Klassifikation von Überwachungsansätzen

Es existiert eine Vielzahl von Ansätzen zur Überwachung von Dienstgütemerkmalen in dienstbasierten Architekturen. Als Basis für die weitere Betrachtung existierender Ansätze sowie für die Einordnung des in dieser Arbeit entwickelten Ansatzes wird in diesem Abschnitt ein Klassifikationsrahmen für Überwachungsansätze vorgeschlagen. Der Klassifikationsrahmen erweitert den im Rahmen des EU FP6 IP Projekts SeCSE[7] (Service Centric Systems Engineering) vorgestellten Klassifikationsrahmen (vgl. [Sero5]) um die Präzisierung der die Überwachung durchführenden Parteien sowie die Berücksichtigung der Lokation der Überwachungsinstrumentation.

Der Klassifikationsrahmen ist anhand der nachfolgenden Dimensionen strukturiert:

- Initiator der Überwachung (Wer?)
- Ort der Überwachung (Wo?)
- Zu überwachende Eigenschaften (Was?)
- Zeitpunkt der Überwachung (Wann?)
- Funktionsumfang der Überwachung (Wie?)

Die erste Dimension unterscheidet Überwachungsansätze anhand des Initiators der Überwachung. Eine Überwachung kann durch einen Dienstanbieter, den Dienstnutzer oder eine beliebige dritte Partei initiiert werden. Ebenfalls vorstellbar sind Mischformen, so dass z. B. ein Dienstanbieter seine Dienste selbstständig überwacht; der Dienstnutzer hingegen beauftragt eine dritte Partei mit der Durchführung der Überwachung.

Eng verbunden mit dem Initiator einer Überwachung ist der Ort der Überwachung. An dieser Stelle kann zwischen zentralen und dezentralen Überwachungsansätzen unterschieden werden. Zentrale Ansätze werden immer vom Initiator selbst durchgeführt, wohingegen im dezentralen Ansatz Überwachungseinheiten innerhalb einer dienstbasierten Infrastruktur verteilt sind.

7 http://www.secse-project.eu/ – abgerufen am 14.04.2009.

Sowohl der Initiator einer Überwachung als auch der Ort der Überwachung haben einen großen Einfluss auf die für eine Überwachung zur Verfügung stehenden Informationen und Datenquellen.

Anhand der zu überwachenden Eigenschaften lassen sich Überwachungsansätze in funktionale und nicht-funktionale Überwachungsansätze unterscheiden. Die funktionale Überwachung eines Dienstes stellt die Korrektheit des Dienstes sicher und ist den aus dem Software Engineering vergleichbaren Funktionstests ähnlich. Nichtfunktionale Ansätze überwachen Dienstgütemerkmale verschiedenster Art (siehe Abschnitt 2.3.2), welche als nicht-funktionale Eigenschaften bezeichnet werden. Ebenfalls Bestandteil der zu überwachenden Eigenschaften ist die Unterscheidung in zustandsbasierte und historienbasierte Überwachungssysteme. Zustandsbasierte Systeme reagieren auf Änderungen einzelner Zustandsvariablen, wohingegen historienbasierte Systeme auf der Analyse historischer Protokollierungsinformationen basieren.

Ein weiteres Kriterium stellt der Zeitpunkt der Überwachung dar. Hierbei lassen sich Überwachungsansätze unterscheiden, die in integrierter Form bei der Ausführung eines Systems Daten sammeln und dadurch Bestandteil eines Ablaufs sind. Alternativ kann die Datensammlung von der eigentlichen Systemausführung entkoppelt durchgeführt werden. Die Überwachung selbst ist somit nicht annähernd in Echtzeit in eine Systemausführung integriert, sondern wird mit leichter zeitlicher Verzögerung parallel ausgeführt.

Abschließend können Überwachungsansätze auch anhand des durch sie unterstützten Funktionsumfangs klassifiziert werden, wobei es hierbei eine Vielzahl möglicher Ausprägungen gibt. Zu den wichtigsten Ausprägungen zählen (in Anlehnung an [Ser05]): die Unterscheidung in proaktive und reaktive Überwachungsansätze, wobei bei einer proaktiven Überwachung im Vergleich zur reaktiven Überwachung nicht nur auf festgestellte Abweichungen von Dienstgütevereinbarungen reagiert wird, sondern versuchen vielmehr durch geeignete Verfahren zukünftige Ereignisse frühzeitig zu erkennen und Gegenmaßnahmen einzuleiten. Eng hiermit verbunden ist die Unterscheidung in reine Erkennungsansätze sowie Ansätze, die auch die Diagnose von Ursachen sowie deren Behebung unterstützen. Gleichermaßen ist eine Unterscheidung anhand des Abstraktionsgrades des zu überwachenden Objekts möglich. Man kann z. B. in eine Überwachung auf Geschäftsprozessebene, auf Workflowebene sowie auf Ebene der konkreten Dienste unterscheiden. Zusätzlich kann man Überwachungsansätze anhand des Ursprungs der zu überwachenden Eigenschaften unterscheiden. Diese können einerseits automatisch abgeleitet oder manuell durch einen Workflowdesigner beschrieben werden. Andererseits lassen sich die Überwachungsanforderungen in den Workflow als Aktivitäten integrieren oder aber durch ein externes Überwachungssystem umsetzen. Eine Unterscheidung auf Basis des gewählten Validierungsansatzes ist ebenfalls möglich. Hierbei wird die Methode zur Überprüfung als Klassifikationsmerkmal herangezogen, wie z. B. die Unterscheidung in Verfahren, die auf der Verifikation von Modellen oder aber der Überprüfung von Vor- und Nachbedingungen beruhen. Die Ausdrucksstärke der der Überwachung zu Grunde liegenden Spezifikationssprache ist das letzte der zu nennenden Merkmale im Hinblick auf eine Klassifikation anhand des Funktionsumfangs.

3

VERTEILUNG VON ÜBERWACHUNGS- UND STEUERUNGSEINHEITEN

In diesem Kapitel wird die Verteilung von Überwachungs- und Steuerungseinheiten innerhalb einer dienstbasierten Infrastruktur untersucht. Bevor verschiedene Ansätze zur Verteilung vorgestellt und anschließend evaluiert werden, erfolgt die Einführung grundlegender Konzepte sowie die Klassifikation allgemeiner Lösungsstrategien. Die Verteilung der Überwachungs- und Steuerungseinheiten (engl. Monitoring & Alignment Units – MAU) wird nachfolgend als mathematisches Optimierungsmodell formuliert und unter Verwendung gemischt-ganzzahliger Optimierungsverfahren gelöst. Darüber hinaus werden verschiedene Heuristiken untersucht, die eine Lösung des Optimierungsmodells zur Laufzeit durch autonom agierende Überwachungs- und Steuerungseinheiten mit einer für die praktische Anwendung ausreichend hohen Lösungsgüte ermöglichen.

3.1 PROBLEMSTELLUNG

Dienstbasierte Architekturen werden zum aktuellen Zeitpunkt primär innerbetrieblich eingesetzt um heterogene Systeme auf Basis von Standards zu integrieren. Die Vorteile eines dienstbasierten Systems, wie z. B. die standardisierte Nutzung von Diensten Dritter, kommen hierbei nur teilweise zum Tragen. Gerade die unternehmensübergreifende Nutzung von Diensten Dritter stellt eines der am häufigsten genannten Fernziele bei der Einführung dienstbasierter Architekturen dar (vgl. die Ergebnisse der Marktuntersuchung in [RM08]). Als Hindernis eines solchen unternehmensübergreifenden Ansatzes wird häufig mangelndes Vertrauen in die Dienstanbieter und deren Datenverarbeitung sowie allgemein die schwer zu überwachende Qualität der bezogenen Leistungen genannt.

Möchte man diesen Herausforderungen durch die Implementierung und Einführung eines Überwachungs- und Steuerungssystems als Baustein einer Dienstgütemanagementstrategie begegnen, so muss man dem verteilten und lose gekoppelten Charakter des zu überwachenden Systems nachfolgen. Eine der Kernfragestellungen in diesem Kontext ist die Platzierung der Überwachungs- und Steuerungseinheiten innerhalb eines verteilten dienstbasierten Systems. Großen Einfluss auf die Komplexität dieses Unterfangens haben dabei die unterschiedlichen zu berücksichtigenden Kontrollsphären von Dienstnutzern, -anbietern und Intermediären, die die Sichtbarkeit von zur Überwachung notwendigen Informationen einschränken können.

Zur Adressierung dieser Herausforderungen im Umfeld dienstbasierter Architekturen kann man auf Arbeiten aus unterschiedlichen Bereichen aufbauen. Eine Grundlage bilden hierbei verschiedene Ansätze zur zentralen Überwachung und (teilweise auch) Steuerung von dienstbasierten Systemen (siehe z. B. [BGG04], [SM06], [BGP08] bzw. Abschnitt 5.6). Zu den Grundlagen zählen weiterhin Ansätze zur verteilten Überwachung von Systemen aus dem Bereich des Netzwerkmanagements, die sich

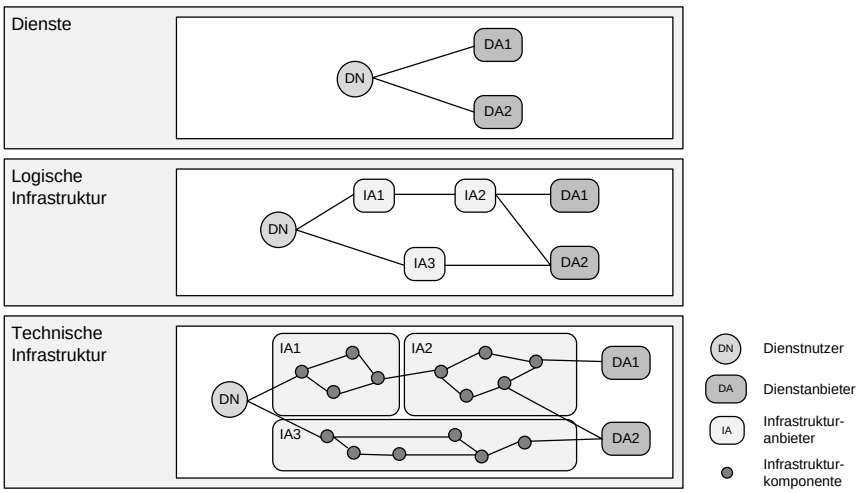


Abbildung 9: Betrachtungsebenen einer dienstbasierten Infrastruktur

teilweise auch mit der Frage nach der Verteilung von Überwachungs- und Steuerungseinheiten innerhalb einer Infrastruktur beschäftigen.

Die Übertragung der Prinzipien des verteilten Netzwerkmanagements auf die in dieser Arbeit zu Grunde gelegten dienstbasierten Szenarien lässt sich für die verschiedenen Merkmale einer SOA unterschiedlich leicht durchführen. Während sich das Merkmal der losen Kopplung von Diensten und den daraus resultierenden (möglicherweise) kurzlebigen Geschäftsbeziehungen zwischen einem Dienstanbieter und Dienstnutzer durch die Instabilität von Routen innerhalb der Verfahren des Netzwerkmanagements modellieren lässt, stellt die durch das SOA-Paradigma geförderte Durchgängigkeit bzw. Transparenz zwischen Fachlichkeit und technischer Umsetzung eine besondere Herausforderung dar.

Analysiert man die Beziehungen zwischen einem Dienstnutzer sowie den Anbietern der durch den Nutzer konsumierten Dienste, so lassen sich verschiedene Betrachtungsebenen dieser Beziehungen identifizieren. Abbildung 9 zeigt eine Aufteilung in drei Betrachtungsebenen, welche durch ein Überwachungs- und Steuerungssystem als Teil des Dienstgütemanagements zur Realisierung der geforderten Durchgängigkeit integriert werden müssen. Auf Dienstebene sind Anforderungen an Dienste sehr abstrakt im Rahmen von Dienstleistungsvereinbarungen bzw. SLA spezifiziert, welche durch die Dienstnutzer (z. B. eine Fachabteilung) selbst erarbeitet werden. In ihnen wird das Verhältnis zwischen den involvierten Parteien direkt geregelt, ohne dass die Vielzahl möglicher zwischengeschalteter Instanzen dabei Berücksichtigung findet. Die Ebene der logischen Infrastruktur integriert hingegen alle betroffenen Instanzen im Rahmen einer solchen Spezifikation. Zumeist existieren in einem Unternehmen auch für das Verhältnis zu Netzanbietern und anderen Intermediären entsprechende Vereinbarungen, die weniger durch eine Fachabteilung als durch die technischen Abteilungen selbst ausgehandelt und verwaltet werden. Die eigentliche Realisation wird auf Ebene der technischen Infrastruktur festgelegt und beinhaltet alle bei einer Dienstnutzung involvierten Instanzen und Komponenten.

Eine Herausforderung liegt somit darin, Anforderungen der Dienstnutzer in die Infrastruktur zu überführen und dabei gleichzeitig Rückkopplungen in die entgegengesetzte Richtung zu berücksichtigen, wobei das verwendete Überwachungs- und

Steuerungssystem in der zur Verfügung stehenden Infrastruktur verteilt sein kann. Eine solche Verteilung von Überwachungs- und Steuerungseinheiten muss auf Grund der unterschiedlichen Kontrollsphären sowohl mit vollständigen als auch unvollständigen Informationen arbeiten können, welche ihrerseits aus SLA oder aus aktuellen Messungen gewonnen werden. Besonders der Effizienz der eingesetzten Verteilungsstrategien kommt in einem verteilten Überwachungs- und Steuerungssystem eine wichtige Rolle zu. Um den verteilten und ressourcenbeschränkten Überwachungs- und Steuerungseinheiten die Ermittlung von Verteilungen innerhalb der durch sie überwachten Kontrollsphären zu ermöglichen, müssen einfache und wenig rechenintensive Verteilungsstrategien gefunden und umgesetzt werden.

3.2 ANWENDUNGSSZENARIO

Auf Basis der in Abbildung 9 dargestellten Geschäftsbeziehungen wird in diesem Abschnitt zur Erläuterung der weiteren Inhalte des Kapitels ein Anwendungsszenario beschrieben, welches die Ausgangsbasis der weiteren Überlegungen bildet. Aus Gründen der Komplexität wird hierbei ausschließlich ein Szenario mit zwei Dienstanbietern diskutiert, dessen Größe zum aktuellen Zeitpunkt auch der in der Unternehmenspraxis vorherrschenden Realität nahe kommt (vgl. [RM08]). Typische in der Literatur diskutierte Szenarien gehen hingegen von Hunderten genutzter Dienste aus.

Das Anwendungsszenario besteht aus den in der Abbildung aufgeführten zwei Dienstanbietern, welche von einem Dienstnutzer nachgefragt werden. Zwischen den Dienstanbietern und dem Dienstnutzer befinden sich drei Infrastrukturanbieter, die jeweils Teile der für die Kommunikation notwendigen Netzwerkinfrastruktur zur Verfügung stellen. Alle involvierten Parteien stellen grundsätzlich mögliche Lokalitäten für eine Überwachung und Steuerung dar, wobei aus Sicht des Dienstnutzers, welcher die Überwachung und Steuerung durchführen möchte, nicht alle Kontrollsphären der Parteien gleichermaßen einsehbar sind. Aus Sicht des Dienstnutzers bestehen Verträge mit zugehörigen Leistungsvereinbarungen mit den Dienstanbietern *1* und *2* sowie den Infrastrukturanbietern *1* und *3*, die von ihm direkt genutzt werden.

Im beschriebenen Szenario stehen dem Dienstnutzer zu Beginn vier Dienstgütevereinbarungen als Grundlage für die Ermittlung möglicher Platzierungen von Überwachungs- und Steuerungseinheiten zur Verfügung. Auf Basis der in Abschnitt 2.3.3 beschriebenen Grundbestandteile von Dienstgütevereinbarungen lassen sich für das Szenario die SLA sowie die darin enthaltenen SLO spezifizieren (siehe Tabellen 1 und 2). Die Inhalte der Tabellen bilden die vertraglich vereinbarten Zielsetzungen in Bezug auf verschiedene Dienstgüteparameter ab. Die betrachteten Parameter innerhalb des Szenarios sind die Antwortzeiten der Dienste, die Verfügbarkeit von Diensten und Infrastruktur sowie der maximal mögliche Durchsatz von Diensten und Infrastrukturkomponenten. Es gilt anzumerken, dass sich die Definitionen des Durchsatzes aus Sicht des Dienstanbieters (Aufrufe pro Minute) und des Infrastrukturanbieters (MBit pro Sekunde) unterscheiden.

Die internen Strukturen auf Seiten der Infrastrukturanbieter sowie die in diesen gültigen internen Leistungsvereinbarungen, die durch so genannte Operating Le-

	Dienstanbieter A	**Dienstanbieter B**
Antwortzeit (ms)	1.000	800
Verfügbarkeit (%)	99,5	99,0
Max. Durchsatz (Anfragen/Minute)	1.500	2.000

Tabelle 1: Zielsetzungen der Dienstanbieter

	Infrastruktur-anbieter 1	**Infrastruktur-anbieter 3**
Verfügbarkeit (%)	99,9	99,5
Max. Durchsatz (MBit/Sekunde)	100	1.000

Tabelle 2: Zielsetzungen der Infrastrukturanbieter

vel Agreements definiert werden (siehe hierzu [Olb08] und [GC07]), können für eine Platzierung zunächst nicht verwendet werden. Eine Platzierung muss somit zu Beginn mit unvollständigen Informationen durchgeführt werden, die zur Laufzeit durch Informationen aus dem Überwachungssystem verfeinert werden kann.

3.3 VORAUSSETZUNGEN UND GRUNDLEGENDE KONZEPTE

In diesem Abschnitt wird zunächst ein Schema zur Klassifikation von Verteilungsstrategien vorgestellt, mit dessen Hilfe nachfolgend unterschiedliche Ansätze leicht miteinander verglichen werden können. Weiterhin erfolgt die Definition des Monitoring Unit Location Problem als Basis für die weiteren Überlegungen in diesem Kapitel sowie die Diskussion von Kosten und Nutzen der Verteilung einer Überwachungs- und Steuerungslösung. Den Abschluss macht die Analyse der notwendigen Datenbasis, auf welche die in dieser Arbeit vorgestellten Verfahren angewendet werden.

3.3.1 *Ein Klassifikationsschema für Verteilungsstrategien*

Bevor in den nachfolgenden Abschnitten mögliche Lösungsstrategien für die Verteilung von Überwachungs- und Steuerungseinheiten entwickelt und evaluiert werden, soll zu Beginn in diesem Abschnitt ein Klassifikationsschema für die Einordnung von Lösungsstrategien präsentiert werden. Das Klassifikationsschema findet seine Anwendung in der Abgrenzung der in diese Arbeit vorgestellten Lösungsstrategien zu anderen Arbeiten aus verwandten Themenfeldern.

Die nachfolgende Aufzählung stellt eine Übersicht möglicher Merkmale zur Klassifikation von Lösungsstrategien dar. Eine Einordnung von Lösungsstrategien kann erfolgen auf Basis:

- des unterstützten *Verteilungsgrads* der Lösung. Man unterscheidet hierbei im Detail zwischen verteilten und zentralen Ansätzen.

- des *Zeitpunkts der Durchführung* der Verteilungsstrategie. Mögliche Ausprägungen in dieser Kategorie sind Verfahren, die zur Entwurfszeit oder zur Laufzeit des Systems angewendet werden.
- der *vorliegenden Informationen* zur Unterstützung der Entscheidungsfindung. Man unterscheidet Ansätze auf Basis unvollständiger sowie vollständiger Informationen.
- der *Art der Verteilung*. Es lassen sich Verfahren unterscheiden, die eine einmalige statische Verteilung vornehmen und solche, die eine dynamische Verteilung unterstützen.
- der unterstützten *Planungsstufen* eines solchen Ansatzes. Man kann einstufige und mehrstufige Entscheidungsverfahren unterscheiden.
- der *Echtzeitfähigkeit* des Ansatzes. Je nach Komplexität des Ansatzes ist dessen Echtzeitfähigkeit auf bestimmte Anwendungsszenarien eingeschränkt.

Auf Grundlage dieser Merkmale lassen sich verschiedene Ansätze vergleichen und auch der in dieser Arbeit entwickelte Ansatz bzw. die zugehörigen Lösungsstrategien einordnen. Eine solche Zuordnung erfolgt in den nachfolgenden Unterabschnitten, wohingegen der Vergleich mit verwandten Arbeiten in Abschnitt 3.7 erfolgt.

3.3.2 *Definition des Monitoring Unit Location Problem*

Die Verteilung von Überwachungs- und Steuerungseinheiten innerhalb einer dienstbasierten Infrastruktur kann als Lokationsproblem formuliert werden, welches sich mathematisch modellieren lässt (siehe hierzu auch Abschnitt 3.5.1). Als Grundlage für die weiteren Überlegungen innerhalb dieser Arbeit wird ein solches Lokationsproblem wie folgt definiert:

Definition (MULP)**.** *Ein Optimierungsproblem, welches sich mit der Lokation einer zu Beginn unbekannten Anzahl von Überwachungs- und Steuerungseinheiten innerhalb einer Infrastruktur beschäftigt, wird als Monitoring Unit Location Problem bezeichnet. Die Lokation erfolgt zum Zwecke der Erfüllung einer oder mehrerer vordefinierter Zielsetzungen sowie unter der Berücksichtigung möglicher Nebenbedingungen.*

Die verschiedenen anwendbaren Zielsetzungen sowie die Nebenbedingungen, die eine konkrete Ausprägung des MULP im Detail spezifizieren, werden in den nachfolgenden Abschnitten hergeleitet.

3.3.3 *Kosten- und Nutzenmodell für die verteilte Überwachung und Steuerung*

Wie im vorherigen Abschnitt angesprochen, ist das Ziel des MULP die Minimierung der gesamten durch die Überwachung und Steuerung aller verwendeten Dienste anfallenden Kosten. In diesem Abschnitt wird aus diesem Grund das dieser Arbeit zu Grunde liegende Kostenmodell entwickelt und diskutiert. Hierbei liegen ausschließlich die Kosten im Fokus der Betrachtung, die aus der Verletzung von Anforderungen an einen Dienst resultieren. Ausgeschlossen sind hierbei die Kosten, die aus der

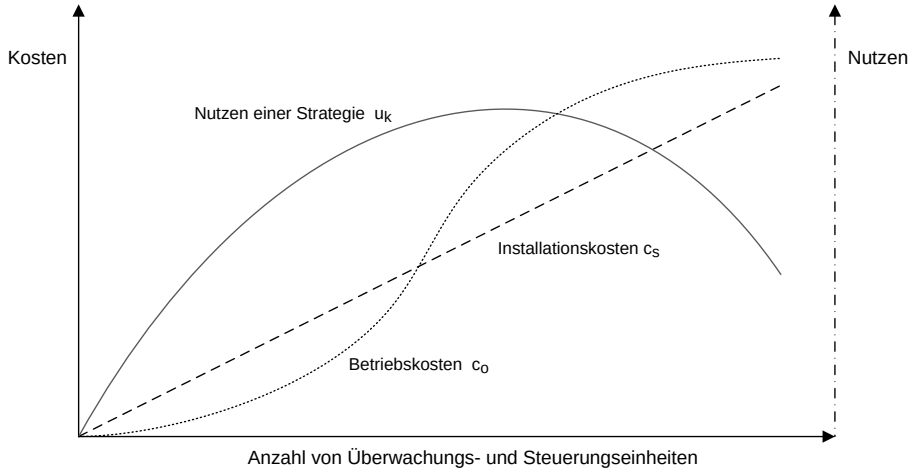


Abbildung 10: Abstrahierte Kosten- und Nutzen-Funktionen

ordnungsgemäßen Nutzung eines Dienstes entstehen, wie z. B. die Kommunikationskosten beim Aufruf eines Dienstes sowie der Zustellung der Antwort des Dienstaufrufs oder aber den Nutzungsentgelten. Implizit enthalten sind hingegen die durch die Überwachung und Steuerung entstehenden zusätzlichen Kosten („Overhead“). Direkt mit dem Kostenmodell verbunden ist das Verständnis des Nutzens einer verteilten Überwachung und Steuerung von Diensten. Aus diesem Grund wird in diesem Abschnitt ebenfalls das zugehörige Nutzenmodell entwickelt, welches einer verteilten Überwachung und Steuerung von Diensten zu Grunde liegt.

Ein Übersicht der Kosten- und Nutzenfunktionen, die der weiteren Arbeit zu Grunde gelegt werden, wird in Abbildung 10 dargestellt und in den folgenden Unterabschnitten erläutert.

Kostenmodell und -funktionen

Bei der Verletzung von Anforderungen durch einen oder mehrere Dienste gilt es verschiedene Gegenmaßnahmen zu initiieren, die eine Wiederherstellung der Einhaltung von Anforderungen ermöglichen. Eine solche Korrekturmaßnahme (engl. alignment) ist je nach Ausprägung mit unterschiedlichen Kosten verbunden. Im Falle des in dieser Arbeit zu Grunde gelegten verteilten Überwachungs- und Steuerungssystems lassen sich grundsätzlich die beiden nachfolgend aufgeführten Kostenarten unterscheiden:

- *Betriebskosten* (engl. operating costs): beinhalten alle Kostenelemente, die zur Laufzeit eines verteilten Überwachungs- und Steuerungsansatzes anfallen (inkl. der notwendigen Kommunikationskosten). Die Betriebskosten werden durch c_o beschrieben.

- *Installationskosten* (engl. setup costs): bestehen aus den für die Installation einer Überwachungs- und Steuerungseinheit auf einem Knoten notwendige Kosten, wie z. B. einer Gebühr zur Platzierung einer MAU bei einem Drittanbieter von Infrastrukturkomponenten. Die Installationskosten werden durch c_s modelliert.

	Orts-abhängige Kosten	Nachfrage-abhängige Kosten	Dienst-abhängige Kosten
Betriebskosten	+	+	+
Installationskosten	+	-	-

Tabelle 3: Zusammenhang von Kostenarten und Kostenkomponenten

Als Ergänzung hierzu können weiterhin die Migrationskosten genannt werden, welche bei der Relokation von Überwachungs- und Steuerungseinheiten entstehen. Je nach Modellierungsansatz lassen sich diese aber durch die beiden zuvor genannten Kostenarten abbilden.

Die Kosten der Platzierung von Einheiten in geschlossenen Kontrollsphären, welche aus dem Verzicht auf mögliche Relokationen der Einheiten außerhalb der Kontrollsphären sowie aus dem Verzicht auf die ausschließlich in den Kontrollsphären vorhandenen Überwachungsdaten entstehen, sind nicht Gegenstand der weiteren Betrachtungen.

Die beiden Kostenarten können ihrerseits in Kostenkomponenten zerlegt werden. Diese Kostenkomponenten lassen sich in die nachfolgenden Kategorien unterteilen:

- *Ortsabhängige Kostenkomponenten*: hierzu zählen alle Kosten, die von der Platzierung einer MAU abhängen, wie z. B. die Menge an Datenverkehr, welche zur Kontaktierung eines Dienstes notwendig ist sowie die Anzahl der durch eine MAU verwalteten Dienste.
- *Nachfrageabhängige Kostenkomponenten*: Kosten, die von der Häufigkeit eines Dienstaufrufs durch einen Dienstnutzer abhängig sind, fallen in diese Kategorie. Hierzu zählt z. B. ebenfalls das Datenaufkommen bei der Dienstnutzung.
- *Dienstabhängige Kostenkomponenten*: alle Kostenanteile, die durch den Dienst selbst bedingt sind, wie beispielsweise die durch den Korrekturbedarf initiierten Kommunikationskosten.

Tabelle 3 zeigt eine Gegenüberstellung von Kostenarten und Kostenkomponenten sowie deren Gewichtung zueinander. Die Kennzeichnung + repräsentiert einen starken Zusammenhang von Kostenart und Kostenkomponenten für diese Kategorie, wohingegen durch − gekennzeichnete Kombinationen einen maximal schwachen Zusammenhang abbilden.

Als Nächstes wird dem Kostenmodell, welches bisher aus Kostenarten und Kostenkomponenten besteht, ein weiterer Bestandteil hinzugefügt. Auf jede der Kostenkomponenten und somit implizit auch auf die Kostenarten wirken eine Reihe von Einflussfaktoren ein, welche auch im Kostenmodell abgebildet werden sollen. Das vollständige Kostenmodell der verteilten Überwachung und Steuerung von Diensten wird in Abbildung 11 dargestellt.

Es existiert eine große Anzahl von Einflussfaktoren, die auf die Kostenkomponenten bzw. Kostenarten einwirken. Nachfolgend sollen die aus Sicht dieser Arbeit wich-

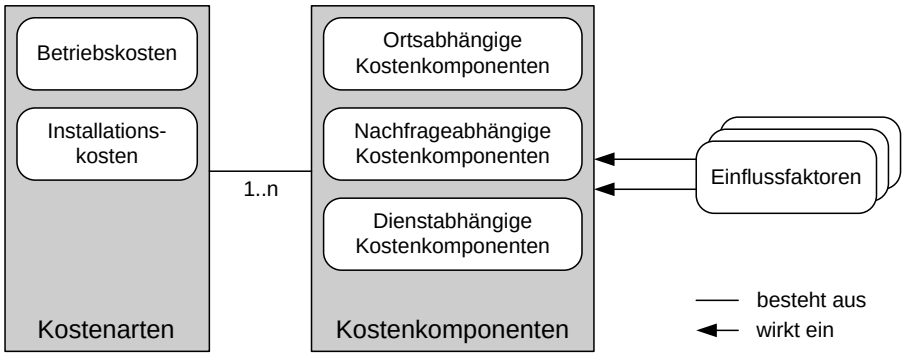


Abbildung 11: Aufbau des Kostenmodells

tigsten Einflussfaktoren kurz aufgeführt werden, wobei die Aufzählung der Einflussfaktoren keinen Anspruch auf Vollständigkeit besitzt. Die Einflussfaktoren können hierbei grundsätzlich stochastischer und deterministischer Natur sein und müssen innerhalb eines mathematischen Optimierungsmodells unterschiedlich berücksichtigt werden.

Zu den für den weiteren Verlauf dieser Arbeit wichtigen Einflussfaktoren zählen das Nachfrageverhalten, welches sich in der Häufigkeit des Aufrufs eines Dienst widerspiegelt ebenso wie die Wahrscheinlichkeit, mit der ein Dienst bei dessen Ausführung korrigiert werden muss. Dieser so genannte Korrekturbedarf basiert auf der Ausfall- bzw. Fehlerwahrscheinlichkeit eines Dienstes. Ebenfalls wichtige Einflussfaktoren sind die Dienstgütemerkmale der Netzinfrastruktur, die verwendet wird, um einen Dienstaufruf bzw. die Korrekturmaßnahmen zu transportieren. Hierzu zählen beispielsweise der Durchsatz eines Netzes genauso wie dessen Antwortzeiten. Die Bearbeitungsdauer eines Dienstes sowie der Korrekturmaßnahme kann genau wie die allgemeine Auslastung eines Dienstes einen wichtigen Faktor bei der Ermittlung der anfallenden Kosten darstellen, ebenso wie die Größe und Anzahl der auszutauschenden Nachrichten oder zu transportierenden Überwachungs- und Steuerungseinheiten. Weiterhin kann man in Bezug auf die Überwachungs- und Steuerungseinheiten die Häufigkeit deren Relokation sowie deren Platzierung als Einflussfaktoren identifizieren. Die Einflussfaktoren bilden einzelne Merkmale des in Abschnitt 2.3.1 in Abbildung 7 eingeführten Qualitätsmodells für dienstbasierte Architekturen ab, insbesondere die Merkmale Verfügbarkeit, Zuverlässigkeit und Leistungsfähigkeit.

Abschließend soll in diesem Abschnitt die den Kostenarten assoziierten Kostenfunktionen, deren jeweiliger Verlauf in Abbildung 10 abstrakt dargestellt ist, diskutiert werden. Die Funktion der gesamten Installationskosten c_s in Relation zur Anzahl der platzierten MAU ist linear, wobei die Steigung der Funktion der Kostensatz für die Installation einer MAU auf einer Infrastrukturkomponente ist. Die Betriebskostenfunktion hingegen zeichnet sich dadurch aus, dass sie zunächst nur leicht ansteigt, in ihrem späteren Verlauf aber ein starkes Wachstum durchläuft, bis sie zu ihrem Ende hin eine gewisse Sättigung erreicht und gegen ein Maximum strebt. Dieser Verlauf lässt sich dadurch erklären, dass zunächst durch Hinzufügen neuer MAU Kommunikationskosten zwischen den MAU untereinander als auch zu deren Managementsystem entstehen, welches durch weiteres Hinzufügen neuer MAU

stark ansteigt (teilweise mit exponentiellem Wachstum). Das Maximum der Betriebskosten ist hierbei durch die Anzahl an Infrastrukturkomponenten definiert, welche mögliche Gastsysteme für MAU darstellen. Es können nicht mehr MAU platziert werden, als Gastsysteme vorhanden sind. Ihr Minimum ist durch die zentrale Lösung definiert, d. h. eine Überwachungs- und Steuerungseinheit, die den gesamten Prozess mit allen Diensten beim Dienstnutzer selbst überwacht.

Nutzenmodell und -funktion

Analog zur Entwicklung des Kostenmodells im vorherigen Abschnitt kann auch ein Nutzenmodell mit zugehöriger Nutzenfunktion definiert werden, welches innerhalb dieses Abschnitts beschrieben wird. Da im Folgenden als Zielsetzung der Optimierung die Minimierung der Gesamtkosten und nicht die Maximierung des Gesamtnutzens im Vordergrund steht, wird das Nutzenmodell nicht mit dem selben Detaillierungsgrad wie das Kostenmodell behandelt. In Abgrenzung zum Kostenmodell werden für das Nutzenmodell ausschließlich die Faktoren betrachtet, die den eigentlichen Nutzen einer verteilten Überwachung und Steuerung ausmachen und nicht wie zuvor deren Einflussfaktoren. Zu den Nutzenfaktoren im Sinne dieser Arbeit zählen (vgl. hierzu auch Abschnitt 2.3.2):

- *Reaktionszeit*: Hauptnutzen der Verteilung von MAU ist die Verbesserung der Reaktionszeit im Falle von Abweichungen vom gewünschten Zustand eines Systems. Dies wird primär dadurch erreicht, dass Abweichungen durch die Nähe der MAU zum Dienst schneller erkannt und durch die durch die MAU direkt durchgeführten Gegenmaßnahmen schnell korrigiert werden können.
- *Skalierbarkeit*: Eine Grenze bei der Verarbeitung von Anforderungsabweichungen bei komplexen dienstbasierten Workflows ist die Leistungsfähigkeit der Workflow-Engine, die einen solchen Workflow ausführt. Sobald die Leistungsgrenzen einer solchen Engine erreicht sind, ist die Behandlung von Abweichungssituationen im normalen Betrieb nicht mehr möglich und sämtliche in Ausführung befindlichen Workflows werden in ihrer Ausführung behindert. Eine Verteilung der Last auf mehrere MAU kann hier Abhilfe schaffen.
- *Robustheit*: Eng mit der Skalierbarkeit einer verteilten Lösung verbunden ist die Anforderung nach Robustheit einer Lösung. Die Verteilung von MAU hilft, den „Single Point of Failure“ bei der Abweichungsbehandlung zu umgehen und die Ausfallsicherheit der Korrekturmethoden durch die Verteilung von MAU zu erhöhen.

Die beiden letztgenannten Nutzenfaktoren lassen sich aber nur begrenzt durch zusätzliche Einführung neuer MAU in eine Infrastruktur verbessern.

Nachfolgend soll die Nutzenfunktion u_k erläutert werden, welche in Abbildung 10 dargestellt ist. Hierbei beschreibt die Funktion u_k die Verteilungsstrategie k für einen gegebenen Workflow. Der Verlauf der konkaven Nutzenfunktion lässt sich dadurch erklären, dass zunächst der Gesamtnutzen durch das Hinzufügen weiterer MAU ansteigt, bis zu dem Punkt, an dem der durch die MAU im System eingeführte Overhead durch zusätzlichen Kommunikationsbedarf und durch die Proxy-Architektur entstehenden Performanzeinbußen den Nutzen der Verteilung wieder teilweise kompensieren. Während Skalierbarkeit und Robustheit durch die Zunahme an MAU

nur teilweise beeinträchtigt werden, so leidet die Gesamtreaktionszeit durch Performanzeinbußen. Der Nutzenzuwachs durch die Verteilung der MAU wird allerdings nicht vollständig kompensiert.

3.3.4 *Untersuchung des Informationsbedarfs und verfügbarer Datenquellen*

Zur Unterstützung der Entscheidung über eine Verteilung von Überwachungs- und Steuerungseinheiten sowie deren konkrete Platzierung innerhalb einer Infrastruktur ist die Analyse verschiedener Datenquellen notwendig, welche in diesem Abschnitt vorgestellt und genauer untersucht werden sollen. Die Datenquellen liefern hierbei die für das in Abschnitt 3.3.3 erläuterte Kostenmodell notwendigen Eingangswerte. Je nachdem, ob die Verteilung der Überwachungs- und Steuerungseinheiten zur Entwurfs- oder zur Laufzeit des Dienstgütemanagementsystems durchgeführt wird, stehen unterschiedlich umfangreiche Datenquellen zur Auswahl. Zunächst soll allerdings der Informationsbedarf bei der Entscheidungsunterstützung bestimmt werden.

Analyse des Informationsbedarfs zur Entscheidungsunterstützung

Grundsätzlich werden für die Bestimmung der Verteilung von Überwachungs- und Steuerungseinheiten drei Typen von Informationen benötigt, auf deren Basis durch die im Folgenden beschriebenen Verfahren Platzierungsentscheidungen getroffen werden. Diese Informationstypen sind:

- Topologieinformationen der Infrastruktur
- Informationen über die Qualitätsmerkmale eines Dienstanbieters
- Informationen über die geplante Nutzung eines Dienstes

Die Topologie gibt vor, wo innerhalb einer Infrastruktur über bestehende Kontrollsphären hinweg die Platzierung von Einheiten grundsätzlich möglich ist. Sie wirkt damit maßgeblich auf die ortsabhängigen Kostenkomponenten des Kostenmodells aus Abbildung 11 ein. Es gilt zu berücksichtigen, dass die Topologieinformationen im Verständnis dieser Arbeit grundsätzlich Informationen aller drei Ebenen der in Abbildung 9 dargestellten Struktur beinhalten. Optimale Entscheidungsgrundlage ist in jedem Fall eine sehr feingranulare Darstellung der Beziehungen, wie sie auf Ebene der technischen Infrastruktur vorliegt, da diese einen maximalen Informationsgehalt besitzt. Inhalte einer solchen Topologiebeschreibung sind auf der einen Seite struktureller Natur, wie beispielsweise die Angabe von Entfernungen der betrachteten Objekte untereinander. Andererseits werden durch sie Dienstgüteeigenschaften der Struktur beschrieben, wie z. B. der Durchsatz von Verbindungen zwischen den betrachteten Objekten. Als Beschreibungsform für diese Informationen können je nach geplanter Verarbeitungsart Adjazenzmatrizen oder -listen gewählt werden.

Während durch die Topologieinformationen die Infrastruktur zwischen einem Dienstanbieter und dem Nutzer des Dienstes beschrieben wird, werden zur Entscheidungsfindung zusätzlich Informationen über die Qualität des Dienstanbieters bzw. des Dienstes selbst benötigt. Diese Informationen stellen die Basis zur Ermittlung der dienstabhängigen Kostenkomponenten des zuvor vorgestellten Kostenmodells

dar. Für die im Weiteren beschriebenen Verfahren wird eine Einschätzung darüber benötigt, mit welcher Wahrscheinlichkeit für einen Dienst eine Korrektur dessen Verhaltens notwendig wird, wobei es hierbei unerheblich ist, ob es sich dabei um einen Fehler oder nur um eine Anforderungsverletzung handelt. Eine solche Einschätzung des Verhaltens eines Dienstes wird als (relativer) *Korrekturbedarf* bezeichnet und wie folgt definiert:

$$\text{Korrekturbedarf} = \frac{\text{Anzahl von Abweichungen und Fehlern}}{\text{Anzahl aller Dienstnutzungen}} \tag{3.1}$$

Weiterhin werden Informationen über den geplanten Einsatz eines Dienstes durch den Nutzer benötigt, die beispielsweise Ergebnisse eines Verfahrens zur Kapazitätenplanung von Diensten bzw. dienstbasierten Workflows sein können. Sie stellen die Eingangswerte der nachfrageabhängigen Kostenkomponenten des Kostenmodells dar. Entsprechende Informationen gehen indirekt in die Berechnung des Korrekturbedarfs ein.[8]

Der Korrekturbedarf stellt die wichtigste Steuerungsgröße für die Verteilung von Überwachungs- und Steuerungseinheiten dar. Seine Berechnung hängt dabei vom Umfang der zur Verfügung stehenden Informationen ab, die im folgenden Unterabschnitt diskutiert werden.

Analyse der vorhandenen Datenbasis

Die Verfügbarkeit von Topologieinformationen als auch von Informationen über die Qualität von Diensten und deren Anbietern hängt von der Form der Geschäftsbeziehungen zwischen den an einer Interaktion beteiligten Parteien ab. Unterschiedliche Kontrollsphären, die in jedem Fall bei einer Interaktion zwischen einem Dienstanbieter und einem Dienstnutzer berührt werden, bedeuten nicht zwangsweise eingeschränkte Sichtbarkeit auf die benötigte Datenbasis. Sämtlicher Informationsbedarf kann auf Basis vertraglicher Vereinbarungen zwischen den Parteien offengelegt werden.[9] In dieser Arbeit wird aber davon ausgegangen, dass nicht alle für die Entscheidung einer Verteilung wünschenswerten Informationen ohne Einschränkung zur Verfügung stehen und somit mit einem gewissen Unsicherheitsfaktor operiert werden muss. Vielmehr lassen sich die folgenden Fälle der Informationsverfügbarkeit unterscheiden:

1. Vollständige Abdeckung des Informationsbedarfs durch zugesicherte Informationen aus Dienstleistungsvereinbarungen.
2. Partielle Abdeckung des Informationsbedarfs durch zugesicherte Informationen, wobei nicht in Dienstleistungsvereinbarungen beschriebene Informationen nicht zur Verfügung stehen.
3. Partielle Abdeckung des Informationsbedarfs durch zugesicherte Informationen, wobei fehlende Informationen durch eigene Überwachungsdaten vervollständigt werden.

8 Für eine Vertiefung des Themas Kapazitätenplanung wird an dieser Stelle auf [ESR+08], [ESN+08] und [EEM+08] verwiesen.

9 Ein Telefoninterview mit dem Leiter Internal Services (Administration) eines multinationalen IT-Dienstleisters im Januar 2008 ergab, dass die Platzierung von Messeinrichtungen in der Kontrollsphäre eines Kunden als Referenzpunkt eine am Markt übliche Vorgehensweise ist.

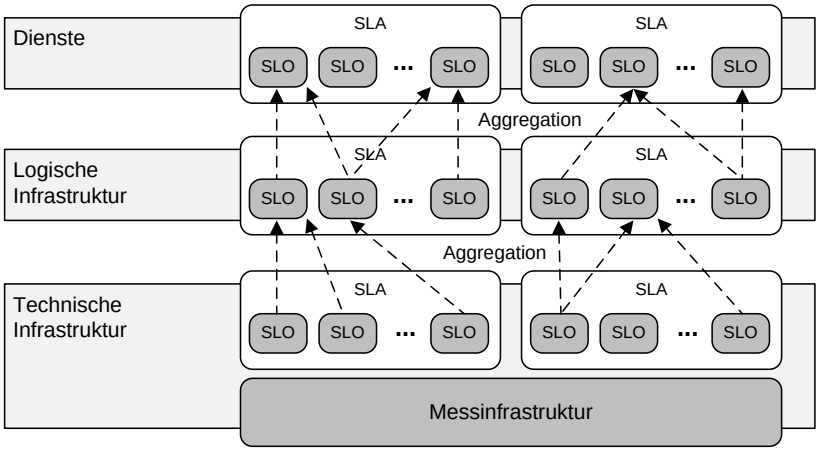


Abbildung 12: Zusammenhang zwischen SLA und SLO

4. Vollständige Abdeckung des Informationsbedarfs durch speziell zu diesem Zweck erhobene Überwachungsdaten.

Zur Sicherstellung einer möglichst breiten Datenbasis zur Ermittlung der Verteilung von Überwachungs- und Steuerungseinheiten wird auf alle der eben beschriebenen Datenbasen zurückgegriffen, wobei die Lösung eines Verteilungsproblems direkt von der verfügbaren Datenqualität und -quantität abhängt.

Nachfolgend wird die Gewinnung der notwendigen Topologieinformationen untersucht. Topologieinformationen auf Ebene der Dienstnutzer sowie auf Ebene der Geschäftsbeziehungen, welche beide Ende-zu-Ende Betrachtungen der Geschäftsbeziehungen darstellen, lassen sich aus den zur Verfügung stehenden Dienstgütevereinbarungen extrahieren. Dabei kann man sich den in Abbildung 12 dargestellten hierarchischen Zusammenhang der SLA und SLO auf den unterschiedlichen Ebenen zu Nutze machen. Sollte für ein Objekt keine Aufgliederung auf der nächst tieferen Ebene vorhanden sein, so muss auf Basis unvollständiger oder unpräziser Informationen die Planung ausgeführt werden. Alternativ können fehlende Informationen zur Laufzeit durch unterschiedliche aktive Messverfahren gewonnen werden. Mögliche Ansätze sind beispielsweise der Einsatz von Internet Control Message Protocol (ICMP) Paketen zur Messung von Laufzeiten zu bekannten Orten im Netzwerk (so genannte Landmarken), wie sie vom *Global Network Positioning* Verfahren von Ng et al. verwendet werden (vgl. [NZ01]). Ähnliche Verfahren beschreiben Francis et al. mit dem *Internet Distance Map Service* (vgl. [FJP+01]) oder Theilmann et al. mit dem *Distance Network Maps* bezeichneten Ansatz (vgl. [TR00]). Die Entwicklung von Verfahren zur Ermittlung der Netzwerktopologie oder die Auswahl existierender Ansätze sind nicht Gegenstand der vorliegenden Arbeit.

Als Nächstes wird in diesem Unterabschnitt die Bestimmung des Korrekturbedarfs eines Dienstes im Detail untersucht. Basis hierfür bildet erneut eine Auswahl der im Qualitätsmodell für dienstbasierte Architekturen (siehe Abbildung 2.3.1) definierten Merkmale. Für den weiteren Verlauf der Arbeit werden die Merkmale „Verfügbarkeit“ (q_{av}) und „Leistungsfähigkeit“ (in Form des Durchsatzes q_{tp} und der Antwortzeit q_{rt}) als Grundlage zur Ermittlung des Korrekturbedarfs ausgewählt, da

es sich bei ihnen um klassische Dienstgütemerkmale handelt, welche sowohl in existierenden Dienstgütevereinbarungen als auch Überwachungssystemen zumeist zur Verfügung stehen. Andere Qualitätsmerkmale sind in ihrer Ermittlung komplexer oder sind als Eingangswerte einer Berechnung nicht verfügbar. Zusätzliche Qualitätsmerkmale können bedarfsweise in die Berechnung des Korrekturbedarfs eingebunden werden.

Wie zuvor beschrieben kann auch die Berechnung des Korrekturbedarfs nicht ausschließlich auf der Verwendung aktueller Messwerte basieren, sondern muss auch die in einer Dienstleistungsvereinbarung zugesicherten Parameter berücksichtigen. Ebenfalls beeinflusst die geplante Nutzungshäufigkeit eines Dienstes als Teil eines dienstbasierten Workflows den absoluten Korrekturbedarf. Zur Bestimmung des Nutzungsverhaltens können zumindest bei erstmaliger Planung nur Schätzwerte zum Einsatz kommen.

Im Detail kann der Korrekturbedarf d_j für einen Dienst j bei erstmaliger Planung wie folgt ermittelt werden, wobei $\hat{h}_j$ den Schätzer für die Anzahl der Dienstaufrufe innerhalb des Planungszeitraums und $\hat{q}^i_{rt}$ den Schätzer für die geplante Antwortzeit eines Dienstaufrufs i definiert. Der Planungszeitraum entspricht zur Vereinfachung dem im SLA definierten Zeitraum, für welchen der angegebene Durchsatz spezifiziert wurde. Für die Bestimmung von $\hat{h}_j$ auf Grundlage der Kapazitätenplanung von Workflows wird an dieser Stelle auf die im vorangegangenen Abschnitt referenzierte Literatur verwiesen.

$$f(\hat{h}_j, q^{SLA}_{tp}) = \begin{cases} \left| q^{SLA}_{tp} - \hat{h}_j \right|, & \text{wenn } q^{SLA}_{tp} - \hat{h}_j < 0 \\ 0, & \text{sonst} \end{cases} \tag{3.2}$$

$$g(i, q^{SLA}_{rt}) = \begin{cases} 1, & \text{wenn } q^{SLA}_{rt} - \hat{q}^i_{rt} > 0 \\ 0, & \text{sonst} \end{cases} \tag{3.3}$$

$$d_j = \tau_1 * (1 - q^{SLA}_{av}) * \hat{h}_j + \tau_2 * f(\hat{h}_j, q^{SLA}_{tp}) + \tau_3 * \sum_{i=1}^{\hat{h}_j} g(i, q^{SLA}_{rt}) \tag{3.4}$$

Die Gleichungen 3.2 und 3.3 beschreiben die Anzahl der durch den jeweiligen Dienstgüteparameter verursachten Abweichungen und Fehler durch den Vergleich des geschätzten Verhaltens in Relation zu den in der Dienstgütevereinbarung beschriebenen Parametern. Weitere Voraussetzung für den in Gleichung 3.4 spezifizierten absoluten Korrekturbedarf ist die Unabhängigkeit der einzelnen Abweichungen bzw. Fehler. Korrelieren diese miteinander, so wird der Korrekturbedarf im Vergleich zur tatsächlichen Situation zu hoch eingeschätzt. Der Parameter τ definiert den Gewichtungsfaktor, mit dem der Einfluss der Verfügbarkeit (τ_1), des Durchsatzes (τ_2) sowie der Antwortzeit (τ_3) auf den gesamten Korrekturbedarf gesteuert werden kann. Es gilt $\sum_{i=1}^{3} \tau_i = 1$.

Die Formulierungen von 3.2 und 3.3 basieren auf der Überlegung, dass sowohl Dienstnutzer als auch -anbieter bei der Formulierung des gemeinsamen SLA entsprechende Reserven einkalkulieren. Anbieter sichern innerhalb des SLA einen durchschnittlichen Erwartungswert für Parameter zu, den sie in jedem Fall erfüllen können.

Temporäre Peaks werden ermöglicht, wie sie durch das kalkulierte Überschreiten von in SLA zugesicherten Obergrenzen für beispielsweise den Durchsatz entstehen. Ebenfalls erlaubt eine vorsichtige Definition benötigter Antwortzeiten auf Seiten des Dienstnutzers eine temporäre Überschreitung der in einem SLA definierten Zusicherungen.

Auf Basis der innerhalb einer Dienstgütevereinbarung festgelegten Parameter sowie den durch einen Kapazitätenplanungsansatz bereitgestellten Aussagen über die Ausprägungen einzelner Parameter können nur die Grenzen für den Korrekturbedarf bestimmt werden. Der Korrekturbedarf wird standardmäßig zu hoch eingeschätzt, da die in den SLA berücksichtigten Reserven sich im Korrekturbedarf widerspiegeln. Im laufenden Betrieb eines verteilten Überwachungs- und Steuerungsansatzes müssen die zu Grunde gelegten Schätzwerte sowie die Werte aus den SLA durch aktuelle Messwerte ergänzt werden.

3.4 HIERARCHISCHE UND PARTIELL-DEZENTRALE VERTEILUNG

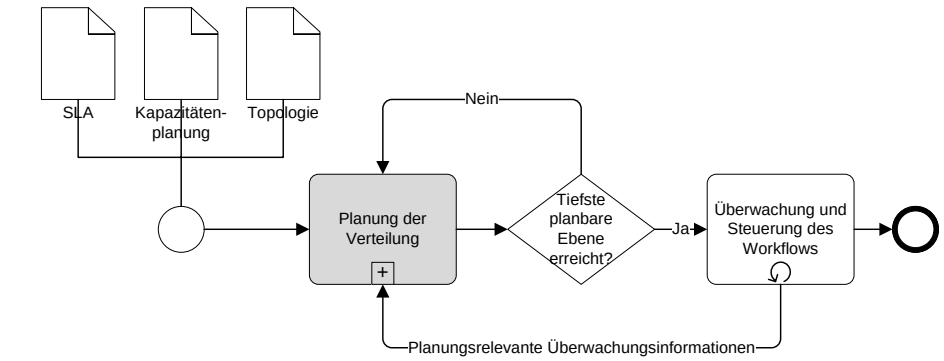
Im folgenden Abschnitt wird ein Ansatz zur Verteilung von Überwachungs- und Steuerungseinheiten beschrieben, welcher auf Grund seines Entwurfs als hierarchischer und zugleich partiell-verteilter Ansatz charakterisiert werden kann. Abschnitt 3.4.1 beschreibt den Aufbau sowie den Ablauf des Verteilungsansatzes. Danach erfolgt in Abschnitt 3.4.2 eine Einordnung des Ansatzes anhand des zuvor vorgestellten Klassifikationsschemas.

3.4.1 *Aufbau und Ablauf des Verteilungsansatzes*

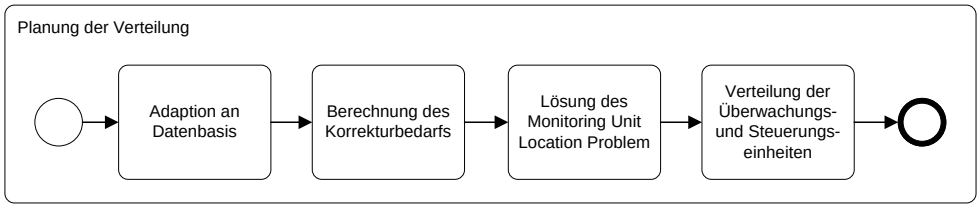
Der Aufbau sowie der Ablauf des dieser Arbeit zu Grunde liegenden abstrakten Verteilungsansatzes ist Gegenstand dieses Abschnitts. Die Herleitung und Erläuterung des Verteilungsansatzes erfolgt auf Basis des zuvor eingeführten Anwendungsszenarios.

Bei Betrachtung des Anwendungsszenarios aus Abschnitt 3.2 kann festgestellt werden, dass die planende Instanz, welche in diesem Fall dem Dienstnutzer entspricht, zu Beginn nicht über alle notwendigen Informationen der einzelnen Kontrollsphären verfügt. Sie kann lediglich auf die in den Tabellen 1 und 2 beschrieben Dienstgütevereinbarungen mit den Infrastrukturanbietern *1* und *3* sowie den Dienstanbietern *1* und *2* zurückgreifen. Eine Planung auf Basis von Detailwissen über den jeweiligen Aufbau der einzelnen Infrastrukturen ist während einer Planung zur Entwurfszeit noch nicht möglich. Die Leistungsmerkmale sowie der Aufbau von Infrastrukturanbieter *2* kann ebenfalls (noch) nicht mit in die Planung einbezogen werden.

Primäre Anforderung bei der Gestaltung des Verteilungsansatzes ist die Gewährleistung der Anwendbarkeit des Ansatzes auf jegliche Form zur Verfügung stehender Informationen. Der Ansatz muss darüber hinaus durch wiederholte Anwendung auf Teile eines Verteilungsszenarios je nach Datenbasis eine Verfeinerung der Verteilung zulassen. Zur Realisierung dieser Anforderungen kommt eine hierarchische Vorgehensweise zum Einsatz, welche es erlaubt ein Verteilungsproblem so zu zerlegen, dass bisher unbekannte Bereiche erst zu einem späteren Zeitpunkt auf Basis neu erhaltener Informationen geplant werden können. Eine Planung für einen solchen Bereich muss nicht von der zentralen Planungsinstanz durchgeführt werden, sondern



(a)



(b)

Abbildung 13: Struktur und Ablauf des Verteilungsansatzes

kann durch eine bereits verteilte Überwachungs- und Steuerungseinheit dezentral vorgenommen werden. Auf das Anwendungsszenario bezogen bedeutet das, dass bei erstmaliger Planung zunächst alle Kontrollsphären der Infrastrukturanbieter als Black-Box betrachtet werden, in welchen keine MAU platziert werden können. Die erstmalige Platzierung erfolgt im Anwendungsszenario zunächst auf Ebene der logischen Infrastruktur. Nach der Verteilung erster MAU sammeln diese weitere Informationen über Topologien und Leistungsmerkmale, die periodisch oder bedarfsweise zur Neubewertung einer Platzierung herangezogen werden können. Eine MAU nimmt demnach zur Laufzeit eine Vermessung ihrer Umgebung vor und kann auf Basis so gewonnener Informationen eine Relokation der eigenen Einheit oder eine zusätzliche Platzierung einer neuen Einheit berechnen. Sie startet als „neuer“ Dienstnutzer das Planungsverfahren für den zu überwachenden Dienst, nur dass dies auf einer tieferen Hierarchieebene mit einem höheren Detaillierungsgrad geschieht. Auf das Anwendungsszenario bezogen bedeutet das, dass die nun bekannten Sphären der Infrastrukturanbieter im Hinblick auf eine Verteilung untersucht werden. Eine Verteilung erfolgt nun auf Basis der technischen Infrastruktur in Abbildung 9.

Zur Lösung des Verteilungsproblems können unterschiedliche Lösungsverfahren zum Einsatz kommen, deren Entwicklung und Evaluierung Gegenstand nachfolgender Abschnitte ist.

Nachfolgend wird der Verteilungsansatz im Detail vorgestellt. Abbildung 13 stellt die Teilprozesse und Aktivitäten des Ansatzes dar. Die Darstellung des Prozesses sowie aller weiteren Prozesse in dieser Arbeit basiert auf der *Business Process Modeling Notation* (BPMN – vgl. [WM08]). Der in Abbildung 13a visualisierte übergeordnete Prozess zeigt die Einordnung der *Planung der Verteilung* im Verhältnis zur nachgeschalteten *Überwachung und Steuerung des Workflows* bzw. Dienstes. Darüber hinaus kennzeichnet die Verzweigung zwischen den beiden Teilprozessen die Möglichkeit

zum Absteigen in tiefere Hierarchiestufen, sofern noch verwendbare Informationen vorliegen.

Die einzelnen Schritte innerhalb des Teilprozesses *Planung der Verteilung* (siehe Abbildung 13b) lassen sich wie folgt beschreiben:

- *Adaption an Datenbasis*: In einem ersten Schritt muss die zur Verfügung stehende Datenbasis ausgewertet und auf dieser Basis entschieden werden, welche Informationen in die weitere Verarbeitung einfließen. Zur Auswahl stehen zu Beginn SLA, Topologieinformationen und Informationen aus der Kapazitätenplanung. Später können Überwachungsdaten hinzugezogen werden.
- *Berechnung des Korrekturbedarfs*: Der Korrekturbedarf wird auf Basis der ausgewählten Informationen berechnet, wobei die Berechnung dem in Abschnitt 3.3.4 beschriebenen Verfahren entspricht.
- *Lösung des Monitoring Unit Location Problem*: Der Korrekturbedarf sowie die Informationen über die Topologie stellen die Eingangsparameter für die Verfahren zur Lösung des MULP dar (siehe Abschnitt 3.5). Nicht zu berücksichtigende Kontrollsphären werden dabei als singuläre Komponenten ohne freie Kapazitäten modelliert.
- *Verteilung der Überwachungs- und Steuerungseinheiten*: Abschließend erfolgt die Verteilung der MAU innerhalb der Infrastruktur unter Verwendung eines geeigneten Frameworks (siehe Kapitel 5 für eine Darstellung des AMAS.KOM Frameworks).

Nachdem die Verteilung berechnet und in der Infrastruktur umgesetzt wurde, kann die eigentliche Überwachung und Steuerung der Dienste bzw. Workflows erfolgen. Die Resultate aus Überwachung und Steuerung haben ihrerseits Einfluss auf die Verteilung. Sollte während der Überwachung festgestellt werden, dass eine bestehende Verteilung nicht mehr ausreichend ist, so kann das Verfahren erneut gestartet werden. Anlässe hierfür können z. B. sein, dass eine platzierte Einheit länger als ein zuvor definiertes Zeitintervall nicht mehr genutzt wurde oder dass die gestiegene Nutzung eines Dienstes eine Relokation einer Einheit notwendig macht. Die erneute Planung auf Basis aktueller Informationen muss dabei nicht für die gesamte Infrastruktur, sondern nur für relevante Teilbereiche durchgeführt werden. Eine erneute Planung kann zusätzlich periodisch gestartet werden, um eine bestehende Verteilung zu bereinigen bzw. zu aktualisieren.

3.4.2 *Klassifikation des Verteilungsansatzes*

Abschließend wird die Einordnung des eben beschriebenen Ansatzes in das in Abschnitt 3.3.1 eingeführte Klassifikationsschema vorgenommen. Der Ansatz lässt sich als partiell-verteilt charakterisieren, da er zu Beginn zwar auf einer zentralen Verteilung durch den Dienstnutzer basiert, diese aber zur Laufzeit wiederholt durch dezentrale Entscheidungskomponenten zur Verfeinerung der Verteilung durchführen lässt. Eine Planung der Verteilung kann zur Entwurfszeit als auch zur Laufzeit erfolgen. Die beiden Zeitpunkte unterscheiden sich lediglich im Umfang der zur Verfügung stehenden Datenbasis. Ist die Datenbasis nur unvollständig, so verringert sich die jeweilige Qualität der Lösung. Der Ansatz ist aber dennoch anwendbar, da er auch mit

Unbekannten operieren kann. Das Verfahren selbst ist dynamisch, wobei zunächst mit einer ersten Verteilung der MAU begonnen wird, welche nachfolgend im Betrieb angepasst werden kann. Die hierarchische Anwendung der Planungsmechanismen macht den Ansatz zu einem mehrstufigen Verfahren, wobei die Mehrstufigkeit aus der wiederholten Anwendung eines an sich einstufigen Ansatzes resultiert. Die Ermittlung einer optimalen Lösung kann dabei nur für die einzelnen Stufen erreicht werden. Je nach Auswahl des eingesetzten Lösungsverfahrens kann der Ansatz annähernd in Echtzeit durchgeführt werden.

3.5 LÖSUNGSVERFAHREN FÜR DAS MONITORING UNIT LOCATION PROBLEM

In diesem Abschnitt soll erläutert werden, wie das zuvor eingeführte Monitoring Unit Location Problem durch Einsatz verschiedener Verfahren gelöst werden kann. In einem ersten Schritt erfolgt hierzu die Modellierung des Problems als mathematisches Optimierungsproblem, welches nachfolgend durch Optimierungsverfahren aus dem Operations Research gelöst wird. Zur Verbesserung des Laufzeitverhaltens des optimalen Lösungsansatzes werden anschließend Heuristiken entwickelt und vorgestellt, die das Problem unter Aufgabe des Optimalitätsanspruches lösen.

Bei der nachfolgenden Formulierung des MULP als mathematisches Optimierungsproblem werden deterministische Größen vorausgesetzt, die durch geeignete Transformation auch aus stochastisch begründeten Größen abgeleitet werden können.

3.5.1 *Modellierung als mathematisches Optimierungsproblem*

Basis für die weitere Betrachtung des Monitoring Unit Location Problem ist dessen Beschreibung als mathematisches Optimierungsproblem. In diesem Abschnitt werden eine mögliche Modellierung vorgestellt sowie die ihr zu Grunde liegenden Annahmen diskutiert (vgl. hierzu auch [13]).

Grundlegende Annahmen für die Modellierung der Verteilung von Überwachungs- und Steuerungseinheiten und somit auch für die Beschreibung des MULP basieren auf der Graphentheorie. Die wichtigsten Annahmen werden nachfolgend aufgeführt:

- Die Geschäftsbeziehungen zwischen Dienstanbietern und -nutzern sowie der weiteren Intermediäre und Netzkomponenten sind als ungerichteter Graph darstellbar.
- Es lässt sich für jede Geschäftsbeziehung bestehend aus einem Dienstanbieter und dem Dienstnutzer ein minimaler Pfad innerhalb des Graphen finden, über welchen die Interaktion mit einem Dienst stattfindet. Als Minimierungskriterium kommen hierbei die Kommunikationskosten zwischen Dienstanbieter und -nutzer zum Einsatz.
- Die Blätter innerhalb eines zwischen den Dienstanbietern und dem Dienstnutzer entstehenden Routingbaums sind die Dienstanbieter, wohingegen die Wurzel des Baums den Dienstnutzer repräsentiert.
- Alle Knoten zwischen den Dienstanbietern und dem Dienstnutzer sowie die Wurzel selbst sind potentielle Kandidaten für die Ausführung einer Überwa-

chungs- und Steuerungseinheit, sofern sie die dafür notwendigen Kapazitäten aufweisen.

Formal werden die Beziehungen zwischen Dienstnutzer und Dienstanbieter sowie der dabei involvierten Intermediäre als ungerichteter Graph $G(V, E)$ definiert, in welchem die Menge V alle Knoten sowie die Menge E alle Kanten beschreiben.[10] Hierbei gilt $E \subseteq V \times V$. Die Menge aller Kandidaten für eine Platzierung von MAU wird durch C beschrieben, für welche gilt: $C \subseteq V$.[11] Für j gilt $j \in S$, wobei $0 \leqslant j \leqslant m$ mit m als Anzahl der durch einen Dienstnutzer insgesamt innerhalb eines dienstbasierten Workflows angefragten Dienste. Ein einzelner Kandidat aus der Menge aller Kandidaten wird durch i beschrieben, wobei $i \in C$ mit $0 \leqslant i \leqslant n$ sowie n die Anzahl aller Kandidaten beschreibt.

Zu den weiteren Parametern eines Modells zur Beschreibung des MULP zählen der Korrekturbedarf d_j eines Dienstes j (vgl. Abschnitt 3.3.4), die Kosten c_i^s und c_{ij}^o (siehe auch Abschnitt 3.3.3) für die Installation einer Überwachungs- und Steuerungseinheit auf Knoten i sowie den Betrieb derselben auf dem Knoten i für den Dienst j. Ebenfalls Berücksichtigung findet eine mögliche Kapazitätsbeschränkung a_i eines Knotens i zur Installation und Ausführung von Überwachungs- und Steuerungseinheiten.

Zur Vorbereitung der Lösung des MULP durch mathematische Optimierungsverfahren werden zusätzlich zwei Entscheidungsvariablen eingeführt, welche von den Lösungsverfahren verwendet werden. Hierbei ist y_i eine binäre Entscheidungsvariable, die beschreibt, ob auf Knoten i eine MAU platziert und ausgeführt wird. Die ganzzahlige Variable x_{ij} bildet den Anteil der Kosten einer Überwachung von Dienst j durch eine MAU auf Knoten i ab.

Eine vollständige Übersicht über die eben eingeführten Modellparameter, welche die Grundlage zur Lösung des mathematischen Optimierungsmodells sowie zur Beschreibung der Heuristiken darstellen, wird in Tabelle 4 wiedergegeben.

Die zuvor eingeführten Parameter reichen aus, um das Monitoring Unit Location Problem in einer deterministischen Form zu beschreiben, welches nachfolgend als deterministisches kapazitiertes MULP bezeichnet wird und die Basis der weiteren Verfahren darstellt. Stochastische Einflussgrößen finden im deterministischen kapazitierten MULP nur in Form der stochastisch erzeugten Modellparameter und Topologien Berücksichtigung.

Als Nächstes gilt es die Ziele der Verteilung von Überwachungs- und Steuerungseinheiten zu definieren. Hierbei lassen sich verschiedene Zielsetzungen unterscheiden, welche nachfolgend aufgeführt werden:

1. Die Minimierung der Reaktionszeit auf eine SLA-Verletzung oder einen Fehler.
2. Die Minimierung der gesamten Kosten, die aus der Überwachung des vollständigen Workflows resultieren.
3. Die Minimierung einzelner Kostenkomponenten, wie z. B. die Kommunikationskosten oder die Installationskosten von MAU.

10 Im Rahmen dieser Arbeit werden die in der Literatur gängigen Abkürzungen V und E verwendet, welche auf den englischen Bezeichnungen *Vertics* und *Edges* für die Knoten- und Kantenmengen basieren.

11 Hierbei stehen die Abkürzungen S für *Services* und C für *Candidates*.

Parameter	Definition
S	Menge aller zur Verfügung stehenden Dienste $S = \{1, ..., m\}$
C	Menge der Kandidaten für die Platzierung von MAU $C = \{1, ..., n\}$
i	Auswahl des Kandidaten i, $i \in C$
j	Auswahl des Dienstes j, $j \in S$
d_j	Korrekturbedarf des Dienstes j
c_i^s	Installationskosten für eine MAU auf Knoten i
c_{ij}^o	Betriebskosten für die Überwachung von j durch i
a_i	Kapazitätsbeschränkung des Knotens i
y_i	Kennzeichnet, ob eine MAU auf i platziert wurde $y \in \{0, 1\}$
x_{ij}	Anteil der Kosten für die Überwachung von j auf i

Tabelle 4: Parameter des Modells

4. Die Minimierung des Bedarfs an kontrollsphärenübergreifender Kommunikation, was ebenfalls der Minimierung der Anzahl von MAU außerhalb der Kontrollsphäre des Dienstnutzers entspricht.

5. Die Maximierung von Skalierbarkeit und Robustheit des Gesamtsystems durch die Nutzung einer verteilten Lösung.

Die eben beschriebenen Zielsetzungen sind nicht vollständig disjunkt und voneinander unabhängig, d. h. einzelne Ziele können sich gegenseitig beeinflussen bzw. bedingen. Der Schwerpunkt dieser Arbeit liegt auf der Erfüllung der Zielsetzungen 1 bis 3.

Weiterhin müssen bei der Zielerreichung eine Reihe von Rahmenbedingungen berücksichtigt werden, welche die zur Anwendung kommenden Lösungsverfahren entsprechend einschränken. Insbesondere zu berücksichtigen sind hierbei die folgenden Aspekte:

- Es existiert nur eine begrenzte Anzahl von Knoten, die die Ausführung von MAU zulassen (Modellierung als Kapazitätsrestriktion der Knoten).

- Eine steigende Anzahl von Überwachungs- und Steuerungseinheiten erzeugt ebenfalls steigende Kosten, die den positiven Effekt einer Verteilung aufheben oder umkehren können.

Im nachfolgenden Abschnitt wird nun auf Basis der eben vorgestellten Modellierungsaspekte ein mathematisches Optimierungsverfahren zur Lösung des MULP beschrieben.

3.5.2 *Ein Ansatz zur optimalen Lösung des MULP*

Die Präsentation eines Verfahrens zur optimalen Lösung des MULP ist Gegenstand dieses Abschnitts. Die Berechnung einer optimalen Lösung für das Monitoring Unit Location Problem stellt für die im Rahmen dieser Arbeit entwickelten Lösungsverfahren einen wichtigen Vergleichsansatz dar. Die Abweichung der Lösung eines solchen Lösungsverfahrens (siehe hierzu Abschnitt 3.5.3) von der optimalen Lösung definiert die so genannte Lösungsgüte des Verfahrens. Ebenfalls wird zur Bestimmung der Verbesserung des Laufzeitverhaltens eines Lösungsverfahrens ein Vergleich mit dem Laufzeitverhalten des optimalen Lösungsverfahrens durchgeführt.

Grundlage zur optimalen Lösung des MULP bildet das so genannte Warehouse Location Problem (WLP)[12], welches im Rahmen des Operations Research dazu verwendet wird, Entscheidungen über die kostenminimale Platzierung von Produktionseinheiten oder Lagerhäusern innerhalb einer bekannten zumeist geographischen Topologie zu treffen. Die Zielsetzung eines WLP ist die Minimierung der Gesamtkosten bei einer solchen Platzierung, wobei die Gesamtkosten aus Installationskosten eines Lagers oder einer Produktionseinheit an einem Ort sowie Transportkosten vom Lager bzw. der Produktionseinheit zu den Kunden des Unternehmens bestehen. Hierbei stellen limitierte Lagerkapazitäten sowie unterschiedliche Bedarfe von Kunden die Rahmenbedingungen einer solchen Optimierung dar. Je nachdem, ob eine stufenweise Belieferung von hierarchisch angeordneten Lagern bzw. Produktionseinheiten zulässig ist, lassen sich einstufige und mehrstufige WLP unterscheiden. Im Falle des mehrstufigen WLP beliefern sich die einzelnen Einheiten untereinander, sofern Bedarfe nicht durch ein einzelnes Lager bzw. eine einzige Produktionseinheit erfüllt werden können. Da der zuvor vorgestellte Verteilungsansatz bereits hierarchisch organisiert ist bzw. arbeitet, reicht zur Abbildung des MULP ein deterministisches einstufiges WLP aus. Ein wichtiges Merkmal des WLP ist, dass schon das deterministische einstufige WLP in seiner unkapazitierten Form (ähnlich wie andere Lokationsprobleme) NP-schwer ist und somit als gemischt-ganzzahliges lineares Programm nur mit hohem Berechnungsaufwand gelöst werden kann. Zur Lösung eines solchen gemischt-ganzzahligen linearen Programms (engl. Mixed Integer Program – MIP) kommen beispielsweise das Branch and Bound Verfahren sowie Lagrange-Relaxationen zum Einsatz, vgl. z. B. [DK97].

Trotz der Komplexität wird das in dieser Arbeit betrachtete Monitoring Unit Location Problem zur Untersuchung in ein Warehouse Location Problem übertragen, da auf dieser Basis eine Vielzahl von etablierten Lösungsverfahren zur Verfügung stehen, auf deren Basis ein effizientes Verfahren zur Lösung des MULP gewonnen werden kann. Darüber hinaus lassen sich die Konzepte und Parameter der beiden Optimierungsprobleme leicht ineinander überführen. Grundvoraussetzung zur Transformation des MULP auf ein WLP ist, dass sich das zu Grunde liegende Problem als bipartiter Graph modellieren lässt. Dies ist im Falle des MULP per definitionem gegeben.

Die zu platzierenden Einheiten innerhalb des WLP sind die Produktionseinheiten bzw. Lager, welche innerhalb des MULP in den MAU ihre Entsprechung finden. Eine MAU „beliefert“ ebenfalls einen Nutzer mit einer Dienstleistung, nämlich der

12 Das WLP wird je nach Autor auch als Facility Location Problem (FLP) bezeichnet.

Monitoring Unit Location Problem	**Warehouse Location Problem**
Überwachungs- und Steuerungseinheiten	Lager bzw. Produktionseinheiten
Dienstanbieter	Kunde
Belieferung Dienstanbieter mit Dienstleistung	Belieferung Kunde mit Produkt
Korrekturbedarf des Dienstanbieters	Nachfrage nach Produkt durch Kunden
Knoten der Infrastruktur	Geografische Standorte

Tabelle 5: Zuordnungen MULP auf WLP

Fähigkeit zur Überwachung des betreffenden Dienstes sowie der Behebung von Fehlern und SLA-Verletzungen. Demnach stellt der Kunde innerhalb eines WLP den Dienstanbieter im MULP dar. Jeder Dienstanbieter besitzt einen individuellen Korrekturbedarf, welcher beschreibt, wie oft der Dienst die Dienstleistungen des korrespondierenden MAU in Anspruch nehmen muss. Knoten innerhalb der Infrastruktur stellen die potentiellen Standorte im Sinne eines WLP dar, zwischen denen bei einer Platzierung entschieden werden muss. Darüber hinaus gilt es, bei Annahme eines einstufigen WLP vor Anwendung eines Lösungsverfahrens die nicht notwendigerweise einstufige Topologie durch eine Transformation auf Basis der Berechnung der kürzesten Wege in eine einstufige Topologie zu überführen. Die eben beschriebenen Zuordnungen werden in Tabelle 5 nochmals zusammengefasst.

Weiterhin können die für eine Platzierung interessanten Knoten eine Kapazitätsbeschränkung aufweisen, welche die Fähigkeit zur Aufnahme einer oder mehrerer MAU bzw. zum Erbringen von Überwachungs- und Steuerungsdienstleistungen begrenzt.

Bei der Lösung des WLP stellt die Distanz zwischen allen betroffenen Elementen einen maßgeblichen Parameter dar. Bei einer Umsetzung auf das MULP lassen sich die Betriebskosten als Entsprechung zur Distanz annehmen. Analog zum WLP existieren auch beim MULP Kosten, welche durch die Installation von Überwachungs- und Steuerungskomponenten auf Knoten entstehen, so dass hier keine konzeptionelle Anpassung notwendig ist.

Bei der Abbildung des MULP auf ein WLP und somit dessen Beschreibung als gemischt-ganzzahliges lineares Programm müssen verschiedene zusätzliche Annahmen und Modellierungsprinzipien gewählt werden. Die Knoten innerhalb des Infrastrukturnetzes beschreiben weiterhin die Anbieter, die Nutzer als auch die Intermediäre von Diensten. Sie werden auf Basis der Parameter *Bedarf* (d_j), verfügbare *Kapazität* (a_i) und *Installationskosten* (c_i^s) innerhalb des Modells differenziert (siehe hierzu auch Tabelle 4). Es gelten weiterhin für die Modellierung des MULP als gemischtganzzahliges lineares Programm die folgenden Überlegungen:

- Der Dienstnutzer kommt einmalig im Modell vor und wird durch $c_i^s = 0$ gekennzeichnet. Die Ausführung der Überwachung und Steuerung auf der Prozess-Engine selbst ist immer möglich und ohne zusätzliche Kosten realisierbar.

Modell 1 Monitoring Unit Location Problem als WLP

Zielfunktion

$$\text{Minimiere } F(x,y) = \sum_{i=1}^{n} c_i^s y_i + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}^o x_{ij} \tag{3.5}$$

Nebenbedingungen

$$\sum_{i=1}^{n} x_{ij} = d_j \quad \forall j \in \{1,..,m\} \tag{3.6}$$

$$\sum_{j=1}^{m} x_{ij} \leqslant a_i y_i \quad \forall i \in \{1,..,n\} \tag{3.7}$$

$$y_i \in \{0,1\}, x_{ij} \geqslant 0 \quad \forall i \in \{1,..,n\}, \forall j \in \{1,..,m\} \tag{3.8}$$

- Ein Dienstanbieter wird durch das Vorhandensein von Korrekturbedarfen modelliert $d_j > 0$. Dienstanbieter stellen die einzigen Elemente dar, die einen Korrekturbedarf aufweisen. Die Installationskosten sind beim Dienstanbieter unendlich $c_i^s = \infty$, sofern er selbst keine Überwachungs- und Steuerungskomponenten ausführt.
- Alle sonstigen Infrastrukturkomponenten sind potentielle Intermediäre. Sie werden durch das Vorhandensein von Installationskosten $c_i^s > 0$ und Kapazitätsrestriktionen $a_i > 0$ modelliert. Kontrollsphären, die nicht mit in eine Planung einbezogen werden, besitzen keine verfügbaren Kapazitäten zur Platzierung von MAU, d. h. $a_i = 0$.
- Die Kanten innerhalb des Infrastrukturnetzes stellen die Verbindungen zwischen den Infrastrukturkomponenten dar und werden durch die Existenz von Betriebskosten modelliert $c_{ij}^o > 0$.

Die Einstufigkeit des Modells bedingt, dass im Falle einer Nicht-Korrigierbarkeit des Problems durch eine Überwachungs- und Steuerungskomponente die Kontrolle direkt an die aufrufende Instanz, in diesem Falle den Dienstnutzer, zurückgegeben wird. Eine Delegation zwischen den MAU ist im Modell nicht vorgesehen. Darüber hinaus beinhalten die Installationskosten einer Infrastrukturkomponente implizit einen Kostenanteil für die Rückdelegation der Kontrolle im Falle einer Nicht-Korrigierbarkeit.

Das mathematische Modell, welches aus der Reduzierung des MULP auf ein klassisches WLP entsteht, kann in Anlehnung an [DK97] wie in Modell 1 dargestellt beschrieben werden. Gleichung 3.5 beschreibt die Zielfunktion, die minimiert wird. Die Zielfunktion entspricht der Summe aller durch eine Platzierung anfallenden Installations- und Betriebskosten, d. h. es werden die Gesamtkosten minimiert. Die Nebenbedingung in Gleichung 3.6 sorgt dafür, dass die Korrekturbedarfe aller Diens-

te erfüllt werden. Gleichung 3.7 formuliert ebenfalls eine Nebenbedingung, welche die Kapazitätsrestriktion der Infrastrukturkomponenten in das mathematische Modell integriert. Es dürfen nur MAU platziert werden, sofern die Kapazität der Infrastrukturkomponente noch ausreichend ist. Das Modell sieht demnach die Berücksichtigung von Kapazitäten vor – soll ein unkapazitiertes Szenario betrachtet werden, so kann Nebenbedingung 3.7 entfallen.

Abschließend gilt es anzumerken, dass in der vorliegenden Formulierung des Modells grundsätzlich die Möglichkeit besteht, dass zwei Überwachungs- und Steuerungskomponenten anteilig die Überwachung und Steuerung eines Dienstes übernehmen können. Um eine eindeutige Zuordnung von Diensten zu MAU sicherzustellen, muss das zu untersuchende Szenario entsprechend angepasst werden. Eine geeignete Auswahl der Kapazitäten im Verhältnis zu den jeweiligen Korrekturbedarfen innerhalb eines Szenarios stellt sicher, dass eine Erfüllung des Korrekturbedarfs durch eine MAU alleine möglich ist.

3.5.3 *Heuristische Ansätze zur Lösung des MULP*

In diesem Abschnitt werden verschiedene Heuristiken diskutiert, welche das zuvor beschriebene MULP lösen. Per definitionem versucht eine Heuristik einen Kompromiss zwischen der Lösungsgüte eines Verfahrens und dem dafür notwendigen Aufwand zu finden. Während das zuvor beschriebene Lösungsverfahren zu einer optimalen Lösung des MULP führt, versuchen die nachfolgend dargestellten Heuristiken „möglichst“ gute Näherungen an eine optimale Lösung des MULP zu finden, ohne dabei die Rechenzeit des optimalen Verfahrens zu benötigen.

Es existiert im Bereich des Operations Research eine Reihe von heuristischen Verfahren zur Lösung von WLP (vgl. [DD95]). Die im Hinblick auf die erbrachte Lösungsqualität sowie der dafür benötigten Laufzeit als effizient bekannten Heuristiken lassen sich primär in zwei Kategorien unterteilen. Verfahren aus der Kategorie der Lagrange-Heuristiken arbeiten unter Verwendung von Lagrange-Relaxationen, welche die Anzahl der zu berücksichtigenden Nebenbedingungen des zu untersuchenden linearen Optimierungsproblems reduzieren (vgl. z. B. [XX08]) und danach mit Hilfe eines Solvers gelöst werden. In die zweite Kategorie fallen Heuristiken, welche multiple Lösungsverfahren miteinander koppeln (vgl. z. B. [CS04]). Hierbei werden beispielsweise Kombinationen aus linearer Programmierung, zufallsbasiertem Runden sowie komplexen Dekompositionsverfahren eingesetzt (vgl. [STA97]).

Die Verfahren beider Kategorien haben gemeinsam, dass zu ihrer Umsetzung entsprechend umfangreiche Hard- und Softwareressourcen benötigt werden, z. B. hochspezialisierte Solver für lineare Programme und umfangreiche Optimierungsbibliotheken. Eine Ausführung auf ressourcenbeschränkten und autonom agierenden Überwachungs- und Steuerungseinheiten ist für beide Kategorien nicht möglich, wenn auf die Einbindung externer Optimierungsdienste verzichtet werden soll.

Bevor die im Rahmen der Arbeit entwickelten Heuristiken im Detail vorgestellt werden, sollen ergänzende Definitionen und Abkürzungen eingeführt werden, die bei der Beschreibung der Heuristiken Verwendung finden.

Parameter	Definition
R	Menge aller Geschäftsbeziehungen eines Nutzers
R_j	Auswahl einer Geschäftsbeziehung, $R_j \subseteq R$
C_j	Menge der Kandidaten für MAU für Dienst j
P_j	Finale Politik für Dienst j
P_{SC}	Politik für vollständiges Szenario SC
$\hat{P}_{SC}$	Anzahl der MAU für vollständiges Szenario SC
c^p_{ij}	Kosten der Politik für Dienst j bei Dienstnutzer i
c^p_{SC}	Kosten der Platzierung für vollständiges Szenario SC
T_{ij}	Menge aller möglichen Platzierungen von i für Dienst j

Tabelle 6: Zusätzliche Parameter für die Heuristiken

Die Menge R bildet alle Geschäftsbeziehungen eines Dienstnutzers zu allen für ihn relevanten Dienstanbietern ab.[13] Sie ist auf Seiten der Infrastruktur durch die Menge der bei einer Dienstnutzung traversierten Knoten und Kanten definiert ($R \subseteq E$). R_j kennzeichnet die konkrete Geschäftsbeziehung (bzw. die hierdurch betroffenen Elemente innerhalb der Infrastruktur) bei der Nutzung von Dienst j. Hierbei gilt weiterhin $R_j \subseteq R$. Die Menge der innerhalb eines Aufrufs von Dienst j traversierten Knoten, die gleichermaßen Kandidaten für eine Platzierung von Überwachungs- und Steuerungseinheiten sein können, wird durch C_j spezifiziert.

Das Ziel der Heuristiken ist es, eine Verteilungsstrategie bzw. konkrete Platzierung für ein gegebenes Szenario zu finden. Ein solches Szenario lässt sich durch eine Kombination von S und C beschreiben. Das Ergebnis jeder Heuristik beinhaltet neben einer konkreten Verteilungspolitik P_{SC} die Anzahl der platzierten MAU $\hat{P}_{SC}$ sowie die Kosten der getroffenen Platzierung c^p_{SC}. Hierbei gilt $P_{SC} = \bigcup_{j \in S} P_j$. P_j beschreibt die Verteilungsstrategie für einen einzelnen Dienst j. Analog können die Gesamtkosten aller Platzierungen innerhalb des Szenarios beschrieben werden: $c^p_{SC} = \sum_{j \in S} c^p_{ij}$, wobei c^p_{ij} die Kosten für einen beliebigen Knoten $i \in C$ repräsentiert, auf dem im Szenario eine MAU platziert wurde. Abschließend soll T_{ij} als Menge aller möglichen Verteilungspolitiken für einen Dienst j definiert werden, wobei $i \in C_j$, d. h. alle Kandidaten innerhalb einer Geschäftsbeziehung berücksichtigen. Es gilt, dass P_j das Minimum der Menge T_{ij} kennzeichnet.

Eine vollständige Übersicht über die eben eingeführten Definitionen und Abkürzungen bietet Tabelle 6.

Nachfolgend werden zwei aufeinander aufbauende Heuristiken zur Lösung des MULP entwickelt und diskutiert. Hierbei handelt es sich um ein Eröffnungsverfahren, welches eine erste möglichst gute Lösung generiert, sowie ein diese Lösung verwendendes Verbesserungsverfahren. Die beiden Heuristiken lassen sich wie folgt charakterisieren:

13 Die Abkürzung R leitet sich vom englischen Begriff *Relationship* ab.

- *Eröffnungsverfahren TPS_MULP*: Die *Tree-based Primary Solution Heuristic* verwendet einen gierigen Ansatz, der durch Einsatz einer die Topologie berücksichtigende Tiefensuche für einzelne R_j die jeweils optimale Lösung findet. Hierbei handelt es sich üblicherweise um das lokale Optimum, welches in einzelnen Fällen auch das globale Optimum darstellen kann. Das Verfahren arbeitet auf einem Teilgraph (Baum) des Infrastrukturgraphen, welcher durch das Routing vom Dienstnutzer zu den Anbietern der Dienste definiert ist. Dieser Teilgraph entspricht R. Die Wiederverwendung von MAU für verschiedene Dienste wird nur innerhalb der Menge der lokalen Optima unterstützt.

- *Verbesserungsverfahren SAIS_MULP*: Die *Simulated Annealing-based Improved Solution Heuristic* versucht das Ergebnis des Eröffnungsverfahrens durch Akzeptanz temporärer Verschlechterungen der lokalen Optima zu verbessern. Bestehende Lösungen einzelner R_j sowie auch neue Lösungen außerhalb des Routingbaums werden miteinander kombiniert, um Kosteneinsparungen durch die Wiederverwendung von MAU für verschiedene Dienste zu realisieren. Die Heuristik arbeitet auf Basis der Simulated Annealing Metastrategie aus dem Operations Research, um den zu durchsuchenden Lösungsraum sinnvoll einschränken zu können (vgl. [DD98]).

Als Grundlage der Diskussion der Funktionsweise der einzelnen Heuristiken in den folgenden Unterabschnitten dient das nachfolgende Beispielszenario. Aus Komplexitätsgründen werden keine Kosten und Kapazitätsbeschränkungen für die Knoten und Kanten innerhalb der Beispieltopologie angegeben. Die Ausgangssituation ist in Abbildung 14a dargestellt. Sie bildet eine einfache Topologie mit 10 Knoten ab. Knoten 0 kennzeichnet in allen in dieser Arbeit diskutierten Problemen den Dienstnutzer. Es gibt zwei Dienstanbieter, welche im Beispiel durch die Knoten 4 und 5 repräsentiert werden. Nicht alle Knoten sind Bestandteil einer Route vom Dienstnutzer zu den Dienstanbietern. Im Beispiel gelten für die Ausgangsituation $S = \{4, 5\}$, $R = \{\{0, 1, 4\}, \{0, 1, 5\}\}$ sowie $C = \{0, 1\}$. Abbildung 14b visualisiert die Situation vor der Anwendung einer Heuristik.

Tree-based Primary Solution Heuristic

Die erste der in dieser Arbeit diskutierten Heuristiken ist ein Eröffnungsverfahren, welches zu einer ersten zulässigen Lösung des MULP führt. Das Verfahren *TPS_MULP* arbeitet auf den Topologieinformationen, die die Routen vom Dienstnutzer zu den Dienstanbietern beschreiben. *TPS_MULP* ist ein gieriges Verfahren, d. h. es wird bei seiner Anwendung auf die Einzelrouten immer die für eine Einzelroute optimale Politik auswählen, ohne dabei das vollständige Szenario zu berücksichtigen. Die daraus resultierende Gesamtpolitik kann optimal sein, muss dies aber nicht. Kosteneinsparungen, welche durch die Wiederverwendung von MAU für unterschiedliche Dienste entstehen, werden nur innerhalb der Menge der lokalen Optima realisiert.

Zur Beschreibung von *TPS_MULP* werden zunächst eine Reihe von Hilfsfunktionen eingeführt. Die Funktion *PRE(r)* ermittelt für einen gegebenen Knoten $r \in R$ dessen Vorgänger in der Route von einem Dienst zu diesem Knoten. Sie wird zur Traversierung einer Route verwendet. Die Funktion *MINIMUM(T_{ij})* ermittelt die Politik mit den minimalen Kosten für ein gegebenes R_j. Die Funktion *NODE(P_j)* gibt

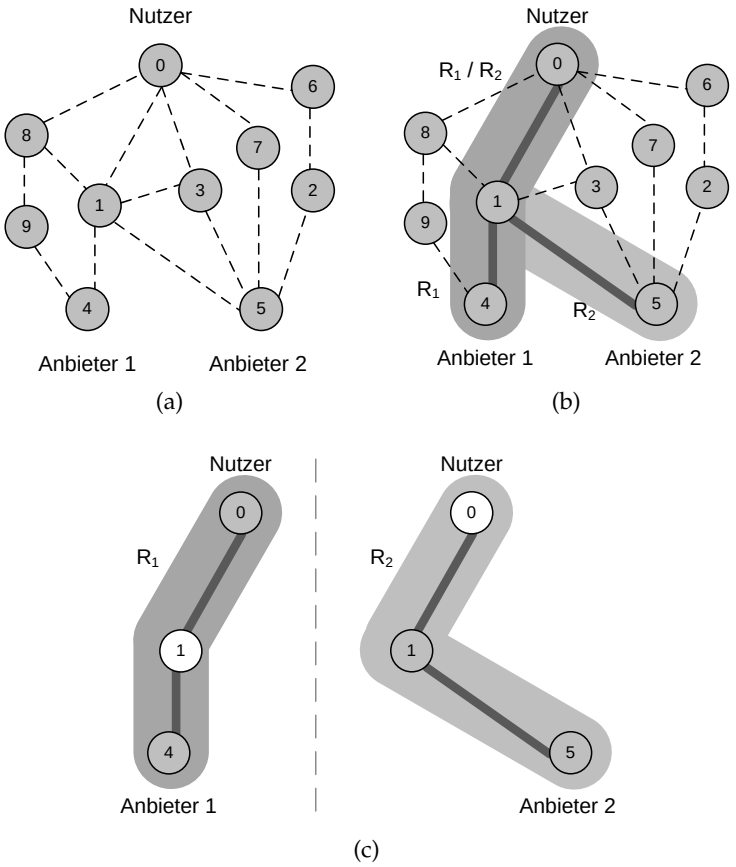


Abbildung 14: Anwendung der Heuristiken: Teil 1

den Knoten zurück, auf dem die MAU für die konkrete Politik P_j platziert wurde und wird zur Unterscheidung von verteilten und zentralen Politiken eingesetzt. Weitere Hilfsfunktionen sind *COST*($\{i, c^p_{ij}\}$), welche die Kosten der Platzierung einer MAU auf Knoten i unter Berücksichtigung einer möglichen Kapazitätsrestriktion zurückgibt, sowie *SAVINGS*(P_{SC}), welche die Einsparungen durch Wiederverwendung von MAU für eine Politik ermittelt.

Die Funktionsweise der Heuristik wird in Algorithmus 2 im Detail beschrieben. Für *TPS_MULP* gilt $C = R \backslash S$. Das Grundprinzip der Heuristik basiert auf der isolierten Traversierung der einzelnen Routen vom Dienstnutzer (Knoten 0) zu den jeweiligen Dienstanbietern. Hierbei wird in jedem Schritt ein fiktiver MAU auf jedem $i \in C_j$ platziert und sowohl die Politik und die Kosten der Politik als Kandidaten für die Gesamtpolitik abgespeichert. Anschließend erfolgt pro Route die Selektion der für diese Route optimalen Politik, d. h. die Politik mit den minimalen Kosten. Die Gesamtpolitik ergibt sich durch die Kombination der lokalen Optima. Abschließend wird ermittelt, ob in der Gesamtpolitik Kosteneinsparungen durch Wiederverwendung von MAU realisiert werden können.

Zur Berechnung einer Politik sowie deren Kosten kommt die Funktion *PLACE*(k, l) zum Einsatz (siehe Algorithmus 3), die auf Basis eines rekursiven Ansatzes zur Tiefensuche die Platzierung für alle Kandidaten auf dem Weg vom Dienstnutzer zum Dienst berechnet. Die Kosten c^p_{kl} einer Platzierung berechnen sich als Summe der Betriebskosten der Lösung sowie den Installationskosten. Ein Großteil der Betriebs-

Algorithmus 2 TPS_MULP(S, C)

Eingabe: S, C
Rückgabe: $P_{SC}, \hat{P}_{SC}, c^p_{SC}$

1: $T_{ij} \leftarrow \{\}, P_j \leftarrow \{\} \quad \forall j \in S, i \in C$
2: $i \leftarrow 0, g \leftarrow 0, h \leftarrow 0$
3: **for all** $j \in S$ **do**
4: $\quad T_{ij} \leftarrow$ PLACE(i, j)
5: $\quad P_j \leftarrow$ MINIMUM(T_{ij})
6: $\quad$ **if** NODE$(P_j) \neq \{0\}$ **then**
7: $\quad\quad g \leftarrow g + 1$
8: $\quad$ **else**
9: $\quad\quad$ **if** $h = 0$ **then**
10: $\quad\quad\quad h \leftarrow h + 1$
11: $\quad\quad$ **end if**
12: $\quad$ **end if**
13: **end for**
14: $P_{SC} \leftarrow \bigcup_{j \in S} P_j$
15: $\hat{P}_{SC} \leftarrow g + h$
16: $c^p_{SC} \leftarrow \sum_{j \in S}$ COST$(P_j) -$ SAVINGS(P_{SC})
17: **return** $\{P_{SC}, \hat{P}_{SC}, c^p_{SC}\}$

Algorithmus 3 PLACE(k, l)

Eingabe: $k \in C, l \in S, C$
Rückgabe: T_{kl}

1: $T_{kl} \leftarrow \{\}$
2: **if** $k \neq l$ **then**
3: $\quad \{$PRE$(k), c^p_{\text{PRE}(k)l}\} \in$ PLACE$($PRE$(k), l)$
4: $\quad c^p_{kl} \leftarrow$ COST$(\{$PRE$(k), c^p_{\text{PRE}(k)l}\}) - c^s_{\text{PRE}(k)} + c^o_{\text{PRE}(k)k} + c^s_k$
5: $\quad T_{kl} \leftarrow T_{kl} \cup \{$NODE$(c^p_{kl}), c^p_{kl}\}$
6: $\quad$ **return** T_{kl}
7: **else**
8: $\quad$ **return** $\{\infty, 0\}$
9: **end if**

kosten besteht aus den Kommunikationskostenanteilen, die aus der zwischen MAU und Dienst notwendigen Kommunikation resultieren (vgl. hierzu auch Abschnitt 3.3.3).

Abschließend soll die Anwendung von *TPS_MULP* auf das zuvor beschriebene Beispiel erläutert werden. Die Anwendung des Verfahrens auf das in Abbildung 14b dargestellte Szenario mit $S = \{4, 5\}$ und $C = \{0, 1, \}$ erzeugt zunächst zwei voneinander unabhängige Suchen nach den lokalen Optima für $C_1 = C_2 = \{0, 1\}$ (siehe Abbildung 14c). Nachdem diese mit $P_1 = \{1, c^p_{11}\}$ und $P_2 = \{0, c^p_{02}\}$ gefunden wurden, können diese zur Gesamtpolitik $P_{SC} = P_1 \cup P_2$ kombiniert werden. Hieraus

resultiert $c^p_{SC} = c^p_{11} + c^p_{02}$ sowie $\hat{P}_{SC} = 2$. Einsparungen durch die Wiederverwendung von MAU sind in diesem Szenario nicht möglich, da die Überwachung und Steuerung der beiden Dienste auf unterschiedlichen Knoten durchgeführt wird.

Simulated Annealing-based Improved Solution Heuristic

Die Heuristik *SAIS_MULP* ist ein Verbesserungsverfahren, dessen Lösungsqualität von der Qualität der Lösung des vorgeschalteten Eröffnungsverfahrens abhängig ist. *SAIS_MULP* baut aus diesem Grund auf dem zuvor beschriebenen *TPS_MULP* auf und verwendet dessen Lösungen zur kontinuierlichen Verbesserung. Die dem Verfahren zu Grunde liegende Idee ist die Kombination bisher isolierter Teilpolitiken zu einer neuen und global besseren Gesamtpolitik. Um dies zu ermöglichen, muss eine Kombination von Politiken durchgeführt werden, d. h. einzelne Politiken werden aufgebrochen und die resultierenden Fragmente mit den Fragmenten anderer Politiken kombiniert. Ausgangsbasis für diese Rekombination ist die Menge aller existierender Einzelpolitiken T_{ij}. Es ist notwendig, dass während des Verfahrens nicht ausschließlich optimale Teilpolitiken P_j gewählt werden, sondern auch temporäre Verschlechterungen bei der Generierung neuer Lösungen zugelassen werden. Eine solche Strategie ist hilfreich, um ein lokales Optimum verlassen und zu einem neuen lokalen Optimum kommen zu können, welches für mehrere j gleichzeitig gültig ist. Die Auswahl einer zunächst schlechteren Lösung wird dann als Teil der Lösungsmenge akzeptiert, wenn durch ihre Selektion die Erzeugung von Kosteneinsparungen möglich ist. Demnach impliziert ein solcher Ansatz die Realisierung von Kosteneinsparungen durch die Wiederverwendung von MAU für verschiedene Dienste.

Zur Umsetzung der temporären Verschlechterung von Lösungen kommt eine angepasste Version des Simulated Annealing (SA) zum Einsatz (siehe z. B. [DD98] und [KGV83]). Der SA-basierte Ansatz wählt in einem ersten Schritt eine zu der aktuellen Lösung P^*_{SC} benachbarte Lösung aus, wobei diese Selektion zufällig durchgeführt wird. Die Funktion zur Bestimmung der Nachbarschaft nutzt dabei ein *m-opt*-Verfahren, welches für jeden Dienst zufällig eine neue Teilpolitik wählt, die von der vorangegangenen Teilpolitik verschieden ist. Die daraus kombinierte neue Lösung P^{New}_G ist im Unterschied zum Eröffnungsverfahren nicht mehr nur auf die Knoten innerhalb des Routingbaums beschränkt. Wenn die Kosten der neuen Lösung niedriger sind als die der alten Lösung, d. h. $c^{New}_{SC} < c^*_{SC}$, so wird die neue Lösung ausgewählt. Sollte die neue Lösung schlechter sein, so wird dennoch mit einer bestimmten Wahrscheinlichkeit die neue Lösung ausgewählt. Die Wahrscheinlichkeit ist dabei abhängig von dem Grad der Verschlechterung sowie einem gesondert zu definierenden Temperaturparameter β, welcher sich im Laufe des Verfahrens schrittweise um den Faktor α „abkühlt“, d. h. gegen Null konvergiert. Zusätzlich erfolgt eine Begrenzung der Iterationszahl des Verfahrens durch den Parameter δ. Das Verfahren terminiert entweder durch Abkühlung oder durch Überschreitung der Anzahl zulässiger Berechnungsschritte.

In Algorithmus 4 wird die Heuristik *SAIS_MULP* in Pseudocode detailliert beschrieben, wobei nachfolgend ausschließlich jene Bestandteile des Verfahrens diskutiert werden, die sich von *TPS_MULP* unterscheiden. Für die Heuristik *SAIS_MULP*

Algorithmus 4 SAIS_MULP$(G, \alpha, \beta, \delta)$

Eingabe: G, α, β, δ
Rückgabe: $P_{SC}, \hat{P}_{SC}, c^p_{SC}$

1: $S, C \in G$
2: $T_{ij} \leftarrow \{\}, P_j \leftarrow \{\} \quad \forall j \in S, i \in C$
3: $i \leftarrow 0, g \leftarrow 0, h \leftarrow 0$
4: **for all** $j \in S$ **do**
5: $\quad T_{ij} \leftarrow$ PLACE(i, j)
6: $\quad P_j \leftarrow$ MINIMUM(T_{ij})
7: **end for**
8: $P_{SC} \leftarrow$ IMPROVE$(\bigcup_{j \in S} P_j, G, \alpha, \beta, \delta)$
9: **for all** $p \in P_{SC}$ **do**
10: $\quad$ **if** NODE$(p) \neq \{0\}$ **then**
11: $\quad\quad g \leftarrow g + 1$
12: $\quad$ **else**
13: $\quad\quad$ **if** $h = 0$ **then**
14: $\quad\quad\quad h \leftarrow h + 1$
15: $\quad\quad$ **end if**
16: $\quad$ **end if**
17: **end for**
18: $\hat{P}_G \leftarrow g + h$
19: $c^p_{SC} \leftarrow \sum_{p \in P_{SC}}$ COST$(p) -$ SAVINGS(P_{SC})
20: **return** $\{P_{SC}, \hat{P}_{SC}, c^p_{SC}\}$

gilt im Gegensatz zu *TPS_MULP* $C = V \backslash S$, d. h. alle Knoten in der Infrastruktur abzüglich der Dienstanbieter selbst sind mögliche Kandidaten für eine Platzierung. Die eigentliche Durchführung des Verbesserungsverfahrens ist Bestandteil der Funktion *IMPROVE*$(P^*_{SC}, G, \alpha, \beta, \delta)$, welche in Algorithmus 5 genauer beschrieben wird. Im Laufe des Verfahrens werden, ebenso wie im Eröffnungsverfahren, eine Reihe von Hilfsfunktionen verwendet, die in einem nächsten Schritt definiert werden.

Zunächst wird die Hilfsfunktion *NEIGHBOR*(G, P^*_{SC}) definiert, welche auf Basis der zufälligen Selektion von Platzierungen für MAU eine neue Lösung für das MULP generiert. Sie bekommt die aktuell beste Lösung P^*_{SC} sowie die gesamte für die Verteilung zur Verfügung stehende Topologie G übergeben und liefert eine neue Lösung P^{New}_{SC} zurück. Eine weitere Hilfsfunktion ist *RANDOM()*, die einen Zufallswert im Intervall $(0, 1]$ generiert und zurückliefert. Die Funktion *TRANSITION*$(c^{Temp}_{SC}, c^{New}_{SC}, \beta)$ berechnet die Wahrscheinlichkeit des Übergangs in einen neuen Zustand auf Basis des Vergleichs der Kosten der alten und neuen Lösung (c^{Temp}_{SC} bzw. c^{New}_{SC}) sowie des aktuellen Temperaturwerts β.

SAIS_MULP führt zunächst, analog zu *TPS_MULP*, die Berechnung sämtlicher möglicher Politiken T und deren Platzierungskosten durch. Hierzu wird ebenfalls die Funktion *PLACE*(k, l) verwendet (siehe Algorithmus 2). Anschließend erfolgt die Auswahl der für den jeweiligen Zweig minimalen Teillösung P_j durch Anwendung der Funktion *MINIMUM*(T_{ij}). Unter Verwendung der Gesamttopologie G, der aktu-

Algorithmus 5 IMPROVE($P^*_{SC}, G, \alpha, \beta, \delta$)

Eingabe: $P^*_{SC}, G, \alpha, \beta, \delta$
Rückgabe: P_{SC}

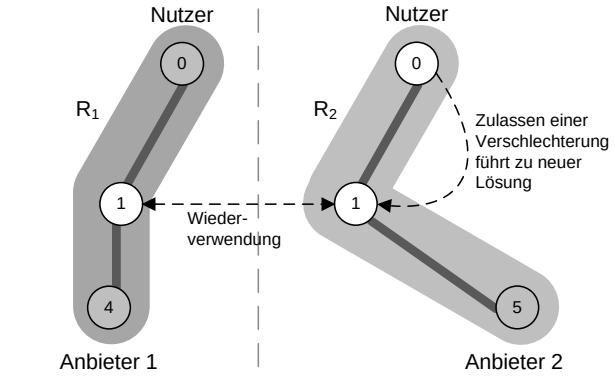
$c^*_{SC} \leftarrow \sum_{p \in P^*_{SC}} \text{COST}(p) - \text{SAVINGS}(P^*_{SC})$
$P^{Temp}_{SC} \leftarrow P^*_{SC}, c^{Temp}_{SC} \leftarrow c^*_{SC}$
$P^{Best}_{SC} \leftarrow P^*_{SC}, c^{Best}_{SC} \leftarrow c^*_{SC}$
$c^{Max}_{SC} \leftarrow c^*_{SC}/2$
$d \leftarrow 0$
while $(d < \delta)$ and $(c^{Temp}_{SC} > c^{Max}_{SC})$ **do**
    $P^{New}_{SC} \leftarrow \text{NEIGHBOR}(G, P^{Temp}_{SC})$
    $c^{New}_{SC} \leftarrow \sum_{p \in P^{New}_{SC}} \text{COST}(p) - \text{SAVINGS}(P^{New}_{SC})$
    if $c^{New}_{SC} < c^{Best}_{SC}$ **then**
        $P^{Best}_{SC} \leftarrow P^{New}_{SC}$
        $c^{Best}_{SC} \leftarrow c^{New}_{SC}$
    end if
    if $\text{TRANSITION}(c^{Temp}_{SC}, c^{New}_{SC}, \beta) > \text{RANDOM}()$ **then**
        $P^{Temp}_{SC} \leftarrow P^{New}_{SC}$
        $c^{Temp}_{SC} \leftarrow c^{New}_{SC}$
    end if
    $d \leftarrow d + 1$
    $\beta \leftarrow \alpha * \beta$
end while
return P^{Best}_{SC}

Algorithmus 6 TRANSITION($c^{Temp}_{SC}, c^{New}_{SC}, \beta$)

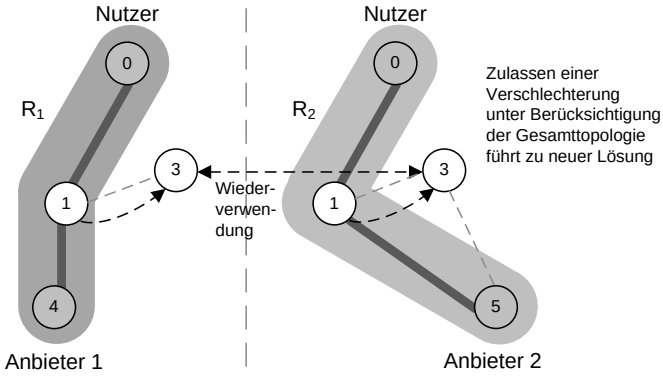
Eingabe: $c^{Temp}_{SC}, c^{New}_{SC}, \beta$
Rückgabe: p^+

$p^+ \leftarrow 0$
if $c^{New}_{SC} < c^{Temp}_{SC}$ **then**
    $p^+ \leftarrow 1$
else
    $p^+ \leftarrow e^{(c^{Temp}_{SC} - c^{New}_{SC})/\beta}$
end if
return p^+

ell besten Lösung $P^*_{SC} = \bigcup_{j \in S} P_j$ sowie der Parameter α, β und δ wird im nächsten Schritt die Funktion *IMPROVE*($P^*_{SC}, G, \alpha, \beta, \delta$) aufgerufen, welche die optimale Politik P_{SC} für das gegebene Szenario zurückliefert. Abschließend wird P_{SC} analysiert und die Gesamtkosten der Platzierung sowie die Anzahl der notwendigen MAU ermittelt, bevor das Verfahren terminiert. Die Platzierung der MAU ist in P_{SC} nicht mehr auf den Routingbaum beschränkt, sondern kann in der gesamten bekannten Topologie vorgenommen werden.



(a)



(b)

Abbildung 15: Anwendung der Heuristiken: Teil 2

Algorithmus 5 bildet das zuvor beschriebene Simulated Annealing ab, wobei die Funktion *IMPROVE*($P^*_{SC}, G, \alpha, \beta, \delta$) zunächst mit einer Lösung auf dem Routingbaum beginnt, diese aber im Laufe des Verfahrens auf die Gesamttopologie ausweitet. Der Start auf einer Lösung mit Platzierungen auf dem Routingbaum stellt einen sehr guten Ausgangspunkt für Verbesserungen dar, da die entsprechenden Lösungen bereits sehr gut sind (siehe hierzu Abschnitt 3.6.3). Wichtiger Bestandteil von Algorithmus 5 ist die in Zeile 4 definierte Grenze für die Verbesserung, welche durch das Verfahren erreicht werden kann. Wenn vor Erreichen der maximalen Anzahl zulässiger Iterationen schon eine Verbesserung zum Eröffnungsverfahren von mindestens 50 % erreicht werden konnte, so terminiert das Verfahren (Zeile 6). Weiterhin hervorzuheben ist Zeile 13, in welcher die Übergangswahrscheinlichkeit definiert wird, mit der auch eine schlechtere Lösung akzeptiert wird. In Zeile 18 des Algorithmus wird die Anpassung der Temperatur durchgeführt.

Die in Algorithmus 6 dargestellte Funktion *TRANSITION*($c_{SC}^{Temp}, c_{SC}^{New}, \beta$) berechnet die benötigte Übergangswahrscheinlichkeit auf Grundlage der Überprüfung des Grads der Verschlechterung unter Einbeziehung der aktuellen Temperatur des Systems. Zeile 5 zeigt die Berechnung der Übergangswahrscheinlichkeit im Fall $c_{SC}^{New} \geqslant c_{SC}^{Temp}$, welche auf dem physikalischen Prozess des Abkühlens von Stahl basiert [KGV83]. Für eine detaillierte Darstellung des physikalischen Prozesses wird zusätzlich auf [MRR+53] verwiesen.

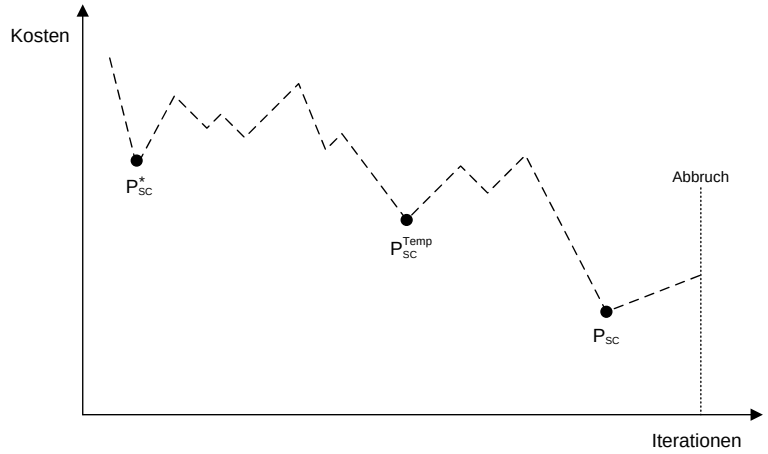


Abbildung 16: Veränderung der Zielfunktionswerte im Beispiel

Abschließend wird an dieser Stelle, analog zur Beschreibung des Eröffnungsverfahrens, die Anwendung des Verbesserungsverfahrens auf das Beispielszenario in Abbildung 14a erläutert. Das Verfahren *SAIS_MULP* wird mit den Parametern G, α, β und δ aufgerufen, wobei für $\alpha = 0.99$, $\beta = 400$ und $\delta = 100$ gelten, d. h. die Startenergie des Systems ist 400, eine Anpassung der Temperatur erfolgt mit dem Faktor 0.99 und es werden maximal 100 Iterationen des Verfahrens zugelassen. Die Wahl der Parameter α, β und δ ist für den Erfolg des Verfahrens kritisch und müssen für das jeweilige Einsatzszenario angepasst werden.

Im Folgenden werden ausgewählte Zwischenlösungen aus dem Verfahren herausgegriffen, um dessen Vorgehensweise zu erläutern. Da Simulated Annealing stark auf dem Einsatz von Zufallsprozessen basiert, werden keine weiteren Zwischenschritte zwischen den dargestellten Lösungen erläutert. In Abbildung 16 wird die Veränderung der Zielfunktionswerte innerhalb des Beispielszenarios grafisch dargestellt.

In einem ersten Arbeitsschritt wird durch Anwendung des Eröffnungsverfahrens eine erste Lösung erzeugt, welche auf dem Routingbaum eine Lösung bestehend aus lokalen Optima generiert. Die Lösung des Eröffnungsverfahrens besteht aus $P^{*}_{SC} = P_1 \cup P_2$ mit $P_1 = \{1, c^{p}_{11}\}$ und $P_2 = \{0, c^{p}_{02}\}$ und $c^{*}_{SC} = c^{p}_{11} + c^{p}_{02}$ (siehe hierzu Abbildung 14b). Danach erfolgt der Aufruf der Funktion *IMPROVE* mit der so entstandenen Lösung zu deren Verbesserung.

Eine Lösung, die den gesamten Zielfunktionswert verbessert, stellt die in Abbildung 15a aufgeführte Lösung dar. In diesem Fall erfolgt nach einer Folge von Iterationen der Übergang auf eine Lösung, welche die Realisation von Kosteneinsparungen durch die Wiederverwendung von MAU zulässt und somit ein weiteres lokales Optimum darstellt. Die Lösung $P^{New}_{SC} = \{\{1, c^{p}_{11}\}, \{1, c^{p}_{12}\}\}$ mit $c^{New}_{SC} = c^{p}_{11} + c^{o}_{12}$ sowie $\hat{P}^{New}_{G} = 1$ platziert die MAU zur Überwachung beider Dienstanbieter auf demselben Knoten. Da das System noch nicht abgekühlt und die maximale Anzahl zulässiger Iterationen noch nicht erreicht ist, wird das Verfahren mit $P^{Temp}_{SC} = P^{New}_{SC}$ und $c^{Temp}_{SC} = c^{New}_{SC}$ fortgeführt.

Nach einer weiteren Folge von Iterationen wird zu einer nächsten Lösung übergangen, die die gesamten Kosten erneut reduziert. Abbildung 15b zeigt die zugehörige Politik. Als Ergebnis des Verbesserungsschritts wird Politik P^{New}_{SC} ausgewählt,

welche eine MAU auf Knoten 3 platziert. Daraus folgt $P_{SC}^{Temp} = P_{SC}^{New}$, $c_{SC}^{Temp} = c_{31}^{p} + c_{32}^{o}$ sowie $\hat{P}_{G}^{Temp} = 1$.

Das Verfahren terminiert nach Erfüllen der Abbruchbedingungen. Da keine Verbesserungen des Zielfunktionswerts mehr möglich waren, stellen $P_{SC} = P_{SC}^{Best} = P_{SC}^{Temp}$, $c_{SC}^{p} = c_{31}^{p} + c_{32}^{o}$ sowie $\hat{P}_{SC} = 1$ das Ergebnis der Heuristik *SAIS_MULP* dar.

3.6 EVALUATION DER LÖSUNGSANSÄTZE

Der folgende Abschnitt beschreibt die Evaluation der in diesem Kapitel entwickelten Ansätze. Zu diesem Zweck wird das Verfahren der Simulation gewählt, da aktuell keine ausreichend großen dienstbasierten Infrastrukturen für Messungen zur Verfügung stehen. Es wird untersucht, wie leistungsfähig die entwickelten Heuristiken sind und in welchen Szenarien die Verteilung von Überwachungs- und Steuerungseinheiten einen Nutzen bringt.

Zunächst werden in Abschnitt 3.6.1 die zu untersuchenden Aspekte beschrieben, auf deren Basis die weitere Evaluation durchgeführt wird. Danach folgt in Abschnitt 3.6.2 eine kurze Beschreibung des Versuchsaufbaus, welcher in seinen technischen Details in Abschnitt A.3 des Anhangs beschrieben wird. Abschließend werden die Ergebnisse der Untersuchungen in Abschnitt 3.6.3 diskutiert.

3.6.1 *Untersuchungsaspekte*

Für die Untersuchungen, welche in diesem und dem nachfolgenden Abschnitt beschrieben werden, gelten die Annahmen aus den vorangegangenen Abschnitten. Im Besondern wird angenommen, dass für die jeweils untersuchten Szenarien vollständige Beschreibungen der Dienstgütevereinbarungen vorliegen. Darüber hinaus sind die jeweiligen Topologien der Infrastrukturen bekannt bzw. lassen sich durch existierende Werkzeuge ermitteln. Weiterhin wird angenommen, dass die Platzierung einer Überwachungs- und Steuerungseinheit in topologischer Nähe zum überwachten Objekt die Reaktionszeit im Falle von Abweichungen und Fehlern verbessert.

Ziel der Untersuchungen ist die Bewertung der Leistungsfähigkeit der Verteilungsstrategien in Form der vorgestellten Heuristiken zur Lösung des MULP. Nebenziel ist es herauszufinden, in welchen dienstbasierten Szenarien eine verteilte Überwachung und Steuerung einen Nutzen verspricht. Es wird analysiert, wie ein optimales Verhältnis von Topologiegröße, Anzahl der Dienstanbieter sowie der zuzuordnenden Anzahl an Überwachungs- und Steuerungseinheiten aussieht. Die Ergebnisse finden als Steuerungsparameter direkten Eingang in das zuvor beschriebene Planungsverfahren.

Die der Untersuchung zu Grunde liegenden Experimente können durch eine Vielzahl von Parametern konfiguriert werden. Die nachfolgend aufgeführten Parameter werden während der Experimente variiert und ihr jeweiliger Einfluss auf das Verhalten des Gesamtsystems analysiert:

- Das Verhältnis von Betriebskosten zu den Installationskosten.
- Das Verhältnis von Dienstanbietern zur Größe der Topologie.

- Das Verhältnis von Kontrollsphären, die nicht eingesehen bzw. innerhalb denen keine MAU platzierten werden können, zur Größe der Topologie.

In Ergänzung zur Variation dieser Steuerungsparameter werden auch die Konfigurationsparameter der synthetisch erzeugten Topologien variiert. Weiterhin ist es möglich, die in Modell 1 definierten Bedingungen einem Szenario anzupassen.

Auf Grundlage dieser Steuerungsparameter lassen sich eine Vielzahl von Szenarien modellieren und untersuchen, wie beispielsweise den Einfluss der Stabilität von Dienstnutzungsbeziehungen auf die Platzierungen von Überwachungs- und Steuerungseinheiten. Die Stabilität kann als Anzahl der gesamten Dienstaufrufe innerhalb eines Szenarios in Relation zur Anzahl der Dienstanbieter mit Diensten vergleichbarer Funktionalität definiert werden. Innerhalb dynamischer und dienstbasierter Workflows kann es jederzeit zu einem Austausch von Dienstanbietern für einen bestimmten Typ von Dienst kommen, was sich in einer Vielzahl von Dienstanbietern im Szenario niederschlägt, welche alle nur selten genutzt werden und jeweils einen relativ geringen Korrekturbedarf aufweisen. Die explizite Untersuchung dynamischer und dienstbasierter Workflows ist nicht Gegenstand der vorliegenden Arbeit.

Die bei der Untersuchung zu Grunde gelegten Infrastrukturen bzw. deren Topologien werden mit Hilfe eines Topologiegenerators erzeugt. Im Detail kommen hierbei synthetische Topologien auf Basis des Verfahrens von Barabasi-Albert (siehe hierzu [BA99] und [AJB99]) zum Einsatz, welche durch eine modifizierte Version des Topologiegenerators BRITE (Boston university Representative Internet Topology gEnerator – vgl. [MMB00]) im Experiment-Testbed erzeugt werden. Eine kritische Diskussion des Barabasi-Albert Ansatzes findet sich bei [CCG+02].

Das Verfahren von Barabasi-Albert basiert im Kern auf zwei Grundannahmen, nämlich den Prinzipien des inkrementellen Wachstums sowie der bevorzugten Verbindung neuer Knoten an Knoten, die bereits eine hohe Konnektivität aufweisen [Had08]. Das Verfahren beginnt mit einer Anzahl m_0 von isolierten Knoten in einer Ebene, die anschließend miteinander durch $m \leqslant m_0$ Links verbunden werden. Hierbei wird eine Seite eines Links zufällig an einem Knoten platziert, wohingegen das andere Ende mit größerer Wahrscheinlichkeit an einen Knoten verbunden wird, welcher bereits stark vernetzt ist [BB03b]. Im Detail wird die Wahrscheinlichkeit $p(i,j)$, mit der ein Knoten i sich mit einem Knoten j verbindet, analog zu Gleichung 3.9 spezifiziert. Variable d gibt den jeweiligen Knotengrad eines Knotens an, wohingegen V die Menge aller Knoten beschreibt, die bereits schon mit der Topologie verbunden sind.

$$p(i,j) = \frac{d_j}{\sum_{k \in V} d_k} \tag{3.9}$$

Das durch BRITE ebenfalls unterstützte Verfahren von Waxman kann auch zur Erzeugung synthetischer Topologien verwendet werden, eignet sich aber für die Abbildung von Internet-ähnlichen Strukturen weitaus weniger als das eben beschriebene Verfahren (siehe [Had08]).[14]

14 In [HPSS03] bilden die Autoren die Netzwerktopologien des amerikanischen Telekommunikationsanbieters AT&T sowie das Deutsche Forschungsnetzwerk unter anderem auf Basis synthetischer Waxman-Topologien ab.

3.6.2 *Versuchsaufbau*

In diesem Abschnitt wird der Versuchsaufbau erläutert, welcher die Grundlage für die Simulationsläufe zur Evaluation der Verteilungsstrategien bzw. der Heuristiken bildet.

Im Rahmen der Simulation werden 9 unterschiedlich parametrisierte Testreihen untersucht, welche jeweils 1.000 Einzelexperimente beinhalten. Im Detail setzt sich eine Testreihe aus jeweils 100 Einzelsimulationen für Topologien mit 10, 20, ..., 100 Knoten pro Testlauf mit derselben Parameterauswahl zusammen. Tabelle 7 zeigt die Parameterauswahl der unterschiedlichen Testreihen. Ziel der Parameterauswahl ist die Überprüfung verschiedener extremer Wertebereiche für einzelne Parameter. So wird schrittweise der Anteil der Dienstanbieter an der Gesamttopologie von anfänglich 20 % auf 50 % gesteigert, wobei die anderen Parameter unverändert bleiben. Gleichermaßen wird das Verhältnis von Installations- zu Betriebskosten von 1:4 auf 2:1 gesteigert sowie der Anteil verbotener Knoten bzw. nicht einsehbarer Kontrollsphären von 0 % auf 50 % gesteigert. Der Korrekturbedarf ist ohne Beschränkung der Allgemeinheit innerhalb einer Testreihe für alle Dienstanbieter gleich.

Die Topologien, welche die dienstbasierten Infrastrukturen repräsentieren, werden wie in Abschnitt 3.6.1 beschrieben durch das Verfahren von Barabasi-Albert für jedes Einzelexperiment zufällig generiert, wobei die Parametrisierung der Topologien innerhalb einer Testreihe zur Gewährleistung der Vergleichbarkeit der Experimente beibehalten werden. Die Kosten werden dabei ebenfalls zufällig gleichverteilt innerhalb eines definierten Kostenintervalls gezogen und den Kanten des Graphen zugeordnet.[15] Zur Generierung der Zufallszahlen kommt ein physikalischer Zufallszahlengenerator zum Einsatz, der auf Basis einer elektronischen Rauschquelle arbeitet.[16] Der weitere Aufbau der Simulationsumgebung *AMAS.KOM Workbench* sowie deren Komponenten werden in Abbildung 31 in Abschnitt A.3 des Anhangs erläutert.

Während der Simulation werden nacheinander für jedes Einzelexperiment der Testreihen die Heuristiken *TPS_MULP* und *SAIS_MULP* aus Abschnitt 3.5.3 sowie der Solver, welcher die optimale Lösung berechnet, ausgeführt und auf Basis der Ergebnisse eine Reihe von Kennzahlen ermittelt. Parallel erfolgt die Anwendung der Heuristik *RND_MULP*, welche eine zufällige Platzierung von Einheiten vornimmt. Auf Grund ihrer schlechten Lösungsgüte wird in dieser Arbeit allerdings auf die Darstellung ihrer Ergebnisse verzichtet.

Die insgesamt während der Experimente ermittelten Kennzahlen sind:

- Absolute Kosten der Heuristiken, des Solvers sowie der zentralen Lösung.
- Absolutes Laufzeitverhalten der Heuristiken und des Solvers.
- Kostenersparnis der Heuristiken und des Solvers im Vergleich zur zentralen Lösung (in Prozent).

15 Eine exemplarische Konfiguration des BRITE-Topologiegenerators findet sich im Anhang dieser Arbeit in Abschnitt A.3.3.

16 Weitere Informationen über den auf [Ric92] basierenden Generator PURAN2 finden sich unter: http://www.puran2.ch/ – abgerufen am 05.04.2009.

Reihe	Anteil Dienstanbieter (%)	Installations- / Betriebskosten	Anteil verbotener Knoten (%)
1	20	1:4	0
2	50	1:4	0
3	20	1:4	20
4	50	1:4	20
5	20	1:4	50
6	20	1:2	20
7	20	1:1	20
8	20	2:1	20
9	50	1:1	20

Tabelle 7: Parametrisierung der durchgeführten Testreihen

- Anzahl der durchschnittlich benötigten MAU in Relation zur Anzahl der zu überwachenden Dienste (in Prozent).
- Lösungsgüte, d. h. der Grad der Annäherung der Heuristiken an den Solver (in Prozent).
- Relative Laufzeit einer Heuristik im Vergleich zum Solver (in Prozent).

Hierbei berechnet sich die Kostenersparnis durch die folgende Formel:

$$\text{Ersparnis} = 1 - \frac{\text{Kosten verteilte Lösung}}{\text{Kosten zentrale Lösung}} \tag{3.10}$$

Die Lösungsgüte wird wie folgt definiert, wobei die Lösung des Solvers die maximal erreichbare Güte darstellt:

$$\text{Lösungsgüte} = \frac{\text{Kosten Solver}}{\text{Kosten Heuristik}} \tag{3.11}$$

Die Kosten definieren sich hierbei als Summe der Betriebs- und Installationskosten für die gesamte zu überwachende Infrastruktur.

Gleichermaßen untersucht wird der Zielkonflikt (engl. Trade-off) zwischen einer angestrebten Verbesserung der Lösungsgüte der untersuchten Heuristiken bei gleichzeitiger Reduktion des dafür benötigten Berechnungsaufwands.

Die Parametrisierung des *SAIS_MULP* Verbesserungsverfahrens wird mit $\alpha = 0.999$, $\beta = 400$ und $\delta = 50$ vorgenommen und in allen in dieser Arbeit beschriebenen Experimenten verwendet. Die Auswahl der Parameter ist das Ergebnis einer Reihe vorangegangener Experimente, die ausgehend von in der Literatur beschriebenen Parametrisierungen für SA-basierte Verfahren an die Anforderungen des MULP angepasst wurde. Die ausgewählte Parameterkombination hat sich als für den Anwendungsfall besonders geeignet erwiesen.

3.6.3 *Diskussion ausgewählter Ergebnisse*

In diesem Abschnitt werden ausgewählte Ergebnisse der Simulation der zuvor spezifizierten Testreihen vorgestellt und diskutiert. Speziell werden in den ersten drei Unterabschnitten die Einflüsse des Anteils an Dienstanbietern, des Kostenverhältnisses als auch des Anteils an verbotenen Knoten auf die Kosteneinsparpotentiale, den Anteil der benötigten MAU, die Laufzeit der Lösungsverfahren sowie die Lösungsgüte der Heuristiken untersucht. Hierbei werden ausschließlich die Ergebnisse der Untersuchung der Topologien mit 100 Knoten diskutiert, da sich die Kennzahlen für ausreichend große Topologien (> 70 Knoten) stabilisieren.

Der Abschnitt schließt mit der Vorstellung und Diskussion der in den vorangegangen Untersuchungen als besonders charakteristisch identifizierten Testreihen. Eine vollständige Darstellung der Ergebnisse aller Testreihen und Topologiegrößen findet sich in Abschnitt A.4 des Anhangs.

Untersuchung der Lösungsgüte in Verbindung mit der relativen Laufzeit

Die Analyse der Lösungsgüte des Verbesserungsverfahrens zeigt, dass dessen Maximum in Testreihe 1 erreicht wird (vgl. Tabelle 8). In diesem Fall wird eine Lösungsgüte von 99,16 % bei einer relativen Laufzeit von 8,77 % erreicht (siehe Tabelle 9). Testreihe 1 repräsentiert eine Topologie mit 20 % Dienstanbietern und keinen verbotenen Knoten sowie dem minimalen Verhältnis aus Installations- und Betriebskosten. Das Minimum der Lösungsgüte von 85,82 % bei einer relativen Laufzeit von 17,11 % tritt in einem stark beschränkten Suchraum auf, welcher in Testreihe 8 durch das maximale Kostenverhältnis sowie jeweils 20 % an Dienstanbietern und verbotener Knoten definiert ist.

Die Untersuchung der relativen Laufzeit des Verbesserungsverfahrens im Vergleich zur Laufzeit des Solver zeigt, dass die beste relative Laufzeit von 3,98 % mit einer Lösungsgüte von 98,97 % in Testreihe 2 mit 50 % Dienstanbietern und 20 % verbotenen Knoten erreicht wird. Das schlechteste Laufzeitverhalten der Heuristik tritt in Testreihe 5 auf, wo bei einer relativen Laufzeit von 24,49 % eine vergleichbare Lösungsgüte von 98,96 % bei dem minimal möglichen Kostenverhältnis erreicht wird.

Insgesamt lässt sich auf Grundlage der Untersuchung feststellen, dass im schlechtesten Falle durch das Verbesserungsverfahren immer noch mehr als 85 % der Lösungsgüte des optimalen Ansatzes erreicht werden kann, wohingegen im ungünstigsten Falle nur knapp ein Viertel der Laufzeit der optimalen Lösung benötigt wird. Würde man ausschließlich das Eröffnungsverfahren zur Anwendung bringen, so könnte die benötigte relative Laufzeit auf 22,45 % unter der Akzeptanz einer minimalen Lösungsgüte von 79,80 % reduziert werden. Die relative Laufzeit hängt direkt vom Anteil verbotener Knoten ab, wohingegen die Lösungsgüte stark durch das Kostenverhältnis beeinflusst wird.

Untersuchung des Einsparpotentials

Die Betrachtung des Einsparpotentials aller Verfahren für Topologien mit 100 Knoten zeigt, dass dessen Minimum in Testreihe 8 erreicht wird (siehe Tabelle 10), in welchem innerhalb des Szenarios das maximale Kostenverhältnis von 2:1 angenommen

Reihe	1	2	3	4	5	6	7	8	9
SAIS	99,16	98,97	99,08	99,11	98,96	96,78	92,14	85,82	89,11
TPS	95,64	91,84	94,99	92,32	94,54	91,04	84,98	80,50	79,80

Tabelle 8: Lösungsgüte bei 100 Knoten (in Prozent)

Reihe	1	2	3	4	5	6	7	8	9
SAIS	8,77	3,98	17,46	18,39	24,49	18,18	18,84	17,11	17,89
TPS	9,65	2,92	17,46	13,79	22,45	16,67	15,94	14,47	11,58

Tabelle 9: Relative Laufzeit bei 100 Knoten (in Prozent)

Reihe	1	2	3	4	5	6	7	8	9
Solver	54,91	49,02	49,21	42,03	40,00	44,62	35,90	28,95	32,68
SAIS	54,53	48,48	48,74	41,51	39,37	42,78	30,43	17,21	24,45
TPS	52,86	44,48	46,54	37,20	36,54	39,17	24,56	11,74	15,63

Tabelle 10: Einsparpotentiale bei 100 Knoten (in Prozent)

wird. Das Verbesserungsverfahren erreicht innerhalb dieses Szenarios eine durchschnittliche Kosteneinsparung von 17,21 %, wohingegen das Eröffnungsverfahren nur eine Kostenreduktion von 11,74 % ermöglicht. Da das optimale Verfahren eine Einsparung von 28,95 % erlaubt und die Lösungsgüte beider Heuristiken in diesem Szenario minimal ist, kann gefolgert werden, dass durch eine angepasste Parametrisierung des Verbesserungsverfahrens eine weitere Annäherung an das optimale Verfahren möglich ist.

Die maximalen Einsparpotentiale finden sich für sämtliche Verfahren in Testreihe 1, d. h. in der Testreihe mit den geringsten Restriktionen. Das Verbesserungsverfahren erreicht hier, analog zum Ergebnis des Solvers, mehr als 54 % an Kosteneinsparung. Auch das Eröffnungsverfahren ist nur rund zwei Prozentpunkte schlechter als die beiden anderen Verfahren.

Die Ergebnisse weisen darauf hin, dass das Einsparpotential direkt von der Restriktionsstärke eines Szenarios abhängig ist. Je mehr Knoten für eine mögliche Platzierung nicht mehr in Betracht kommen, gleichgültig ob diese verboten oder aber Dienstanbieter sind, und je ungünstiger das Kostenverhältnis einer Platzierung im Vergleich zu den Betriebskosten ist, desto geringer fallen die Kosteneinsparung und die Lösungsgüte aus.

Untersuchung des Anteils benötigter MAU

Die Analyse der Testreihen bei 100 Knoten im Hinblick auf den Anteil zu deren Überwachung und Steuerung benötigten MAU identifiziert die Extrema aller untersuchten Lösungsverfahren in den Testreihen 1 und 9 (siehe Tabelle 11). Im schlechtesten Falle werden durch das optimale Verfahren immer noch 65 % der Anzahl der Dienstanbieter als Überwachungs- und Steuerungseinheiten benötigt. Beide Heuris-

Reihe	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Solver	65	40	60	30	45	50	40	25	20
SAIS	70	42	65	32	50	60	45	30	22
TPS	70	42	65	32	50	60	45	30	22

Tabelle 11: Verhältnis MAU zu Dienstanbietern bei 100 Knoten (in Prozent)

tiken errechnen zur Erfüllung derselben Überwachungsaufgaben sogar 70 %. Im besten Falle hingegen kann dieser Anteil auf 20 % (optimale Lösung) bzw. 22 % (Heuristiken) in Testreihe 9 reduziert werden. Dabei stellt Testreihe 9 allerdings ein für die Platzierung von Einheiten stark eingeschränktes Szenario dar. Die topologische Gestaltung dieses Szenarios lässt nur wenig Raum für Einheiten. Betracht man aus diesem Grund das nächst schlechtere Szenario, so findet man dieses mit einer Lösung von 25 % bzw. 30 % in Testreihe 8, welche sich wie zuvor diskutiert durch Berücksichtigung des maximalen Kostenverhältnisses unter allen Testreihen auszeichnet. In diesem Szenario wird eine Platzierung zwar nicht aus topologischer Sicht eingeschränkt, sondern vielmehr durch die vergleichsweise hohen Platzierungskosten beeinflusst.

Eine eindeutige Identifizierung eines Haupteinflussfaktors auf das optimale Verhältnis von Überwachungs- und Steuerungseinheiten zu überwachenden Dienstanbietern ist auf Basis der vorliegenden Testreihen nicht möglich. Allerdings zeigen die Untersuchungsergebnisse, dass mit einer vergleichsweise niedrigen Anzahl von Überwachungs- und Steuerungseinheiten hohe Einsparpotentiale möglich sind. Bei der minimalen Anzahl zu platzierender Einheiten sind dies bei Anwendung des Solvers 32,68 % sowie bei Anwendung des Verbesserungsverfahrens 24,45 %. Bei der maximalen Anzahl der zu platzierenden Einheiten verbessern sich beide Einsparpotentiale auf über 54 %.

Vorstellung und Diskussion weiterer Ergebnisse

Abschließend werden in diesem Unterabschnitt die Ergebnisse der Testreihen 1 und 8 für alle Topologiegrößen vorgestellt. Beide Testreihen stellen auf Grund ihrer Parametrisierung Grenzen für die weiteren Testreihen dar. Testreihe 1 ist dadurch gekennzeichnet, dass ein für die Platzierung günstiges Kostenverhältnis vorliegt sowie wenige Dienstanbieter und keine verbotenen Knoten die Platzierung von Einheiten erschweren. Testreihe 8 weist hingegen ein für die Platzierung teures Kostenverhältnis auf und ist gleichermaßen durch den Anteil an Dienstanbietern und verbotenen Knoten restringiert.

In Abbildung 17 werden die Ergebnisse aus Testreihe 1 visualisiert. Abbildung 17a zeigt den Verlauf der Einsparpotentiale durch die Verteilung der Überwachungs- und Steuerungseinheiten. Diese variiert von anfänglich 38,04 % bei Topologien mit 10 Knoten hin zu einer maximalen Ersparnis von 54,91 % durch die optimale Lösung bei 100 Knoten. Ab einer Topologiegröße von 20 Knoten ist die mögliche Ersparnis bei allen Verfahren größer 45 % und pendelt sich für das Verbesserungsverfahren sowie den Solver ab 80 Knoten auf die maximale Ersparnis ein. Die Entwicklung

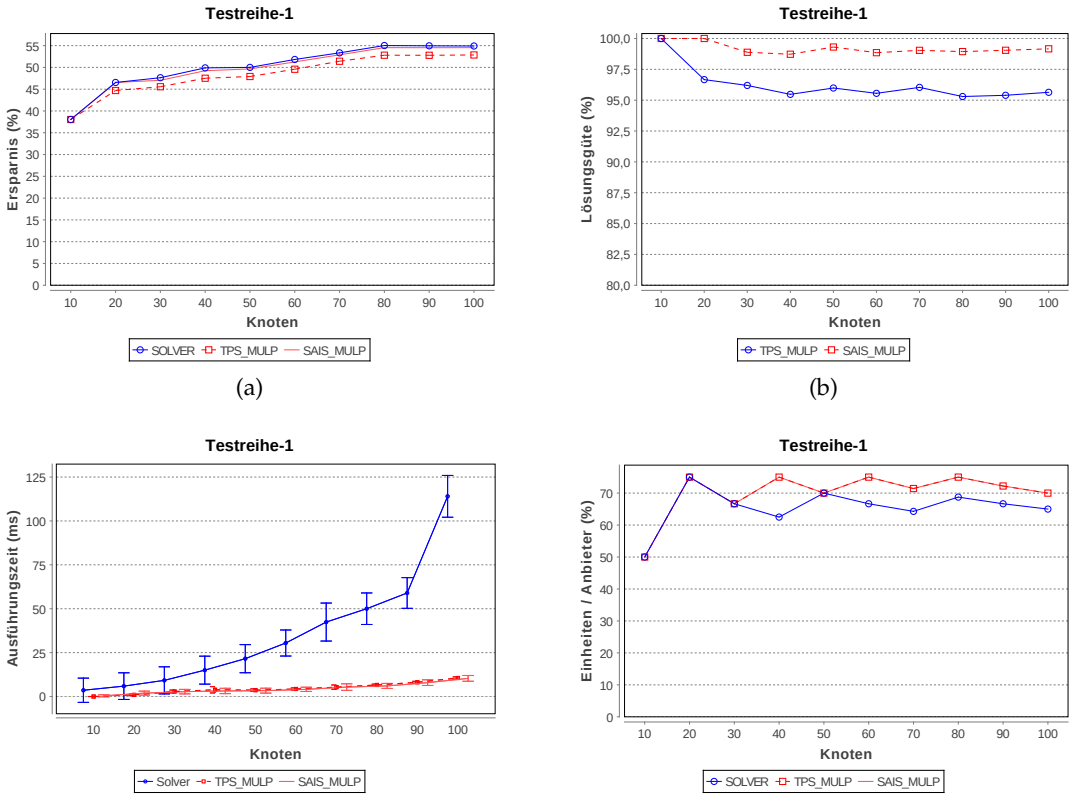


Abbildung 17: Ergebnisse der Testreihe 1

der Lösungsgüte verhalt sich genau entgegengesetzt, wie man Abbildung 17b entnehmen kann. Von anfänglich 100 % Lösungsgüte pendelt sie sich für das Verbesserungsverfahren auf rund 99 % ein – das Eröffnungsverfahren bleibt in sämtlichen Fällen über der 95 % Marke. Die Laufzeit des Solvers in Abbildung 17c zeigt die Charakteristika eines NP-schweren Optimierungsproblems, welche sich bei entsprechender Problemgröße exponentiell entwickelt, wohingegen sowohl das Eröffnungs- als auch das Verbesserungsverfahren einen linearen Verlauf erkennen lassen. Bei der größten Topologie stehen sich hier Laufzeiten von unter 11 ms auf Seiten der Heuristiken und solchen über 110 ms auf Seiten des Solvers gegenüber. Die Auswertung des Verhältnisses von MAU zur Anzahl der Dienstanbieter für die verschiedenen Topologien zeigt, dass diese sich ab der Topologiegröße von 20 Knoten für alle Verfahren zwischen 62 % und 75 % einpendelt. Der Ausreißer bei Topologien mit 10 Knoten lässt sich auf die nicht ausreichende Größe solcher Topologien zurückführen – 50 % entsprechen in diesem Szenario der Platzierung einer MAU und können z. B. eine zentrale Überwachung und Steuerung repräsentieren.

Die Analyse von Testreihe 8 zeigt bei Betrachtung der Einsparpotentiale in Abbildung 18a absolut gesehen wesentlich schlechtere Werte als in der vorangegangenen Testreihe, wenn auch der Verlauf der Kurven vergleichbar ist. Die maximal erreichbaren Ersparnisse sind 28,95 % für den Solver und 17,21 % bzw. 11,74 % für die Heuristiken. Während in Testreihe 1 die Ersparnisse, die von den verschiedenen Verfahren realisiert werden können, noch sehr dicht beieinander lagen, zeigt sich in Testreihe 8

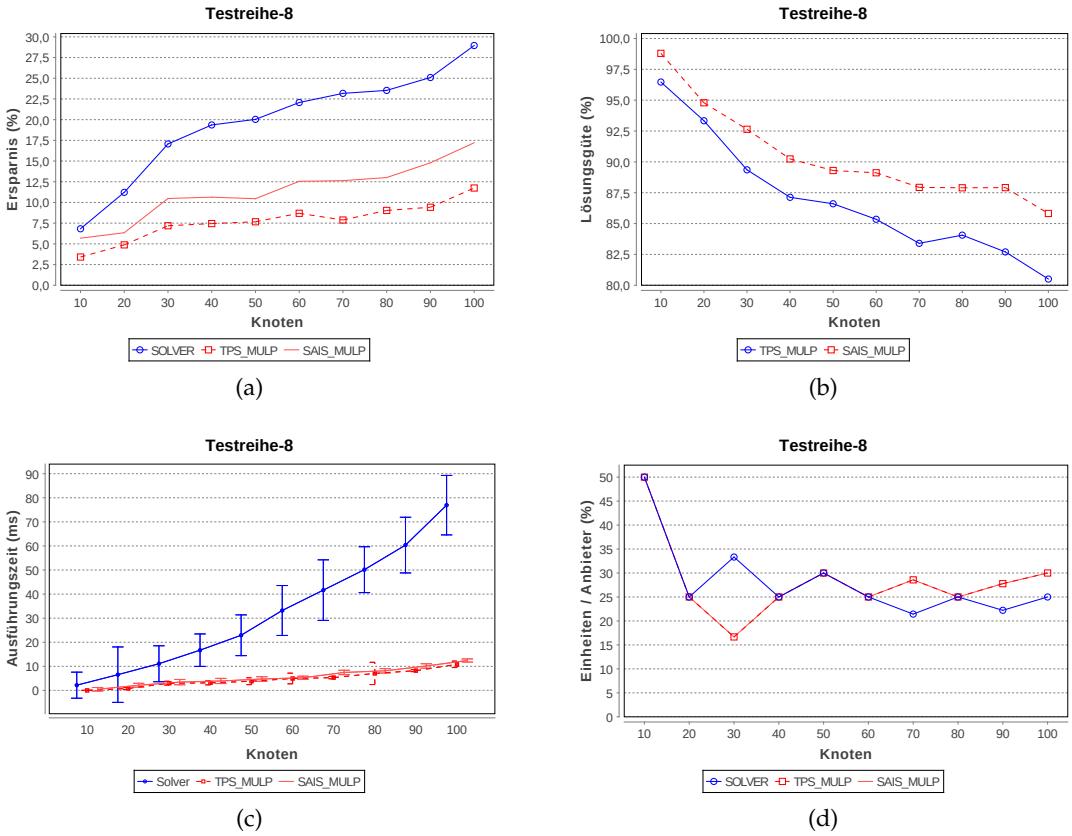


Abbildung 18: Ergebnisse der Testreihe 8

die Überlegenheit des Solvers. Diese Entwicklung spiegelt sich in der in Abbildung 18b dargestellten Lösungsgüte wider. Diese sinkt mit zunehmender Knotenzahl für das Eröffnungsverfahren auf bis zu 80,50 % ab. Bei Betrachtung der Laufzeit in Relation zur Testreihe 1 ist ein vergleichbarer Verlauf erkennbar, auch wenn sich die absoluten Beträge der Laufzeiten unterscheiden (siehe Abbildung 18c). Das Verhältnis der MAU zu der Anzahl der Dienstanbieter, welches Gegenstand von Abbildung 18d ist, pendelt sich, vergleichbar zu dem in Testreihe 1, nach einem anfänglichen Ausreißer innerhalb eines Korridors ein. Im Falle von Testreihe 8 ist dieser durch 16,67 % und 33,34 % begrenzt.

3.7 VERWANDTE ARBEITEN

Nachfolgend werden zu dem in diesem Abschnitt beschriebenen Verteilungsansatz für Überwachungs- und Steuerungseinheiten verwandte Arbeiten vorgestellt. Es lässt sich feststellen, dass keine der untersuchten Arbeiten bisher das in dieser Arbeit zu Grunde gelegte Anwendungsfeld der gleichzeitigen Überwachung und Steuerung berücksichtigt. Ebenfalls ist die Berücksichtigung unterschiedlicher Kontrollsphären nicht Gegenstand der Arbeiten. In jedem Fall ist es aber möglich, Parallelen zwischen dem in dieser Arbeit vorgestellten Ansatz sowie verschiedenen thematisch ähnlichen Arbeiten herzustellen.

Die vorgestellten Arbeiten können zum leichteren Vergleich anhand verschiedener Merkmale unterschieden werden. Das erste Merkmal, mit dessen Hilfe eine solche

Abkürzung	Bedeutung
ÜE	Überwachungseinheiten
P+C	Proxies und Caches
T	Theorie
SC	Set-Cover Problem
FLP	Facility Location Problem
IA	Individueller Ansatz
pM	p-Median/p-Zentrum Problem
H	Heuristik
O	Optimales Verfahren

Tabelle 12: Verwendete Abkürzungen

Unterscheidung möglich ist, ist das *Anwendungsfeld* der Arbeit. Die in diesem Abschnitt vorgestellten Arbeiten teilen sich in die Anwendungsfelder der Lokalisierung von Überwachungseinheiten, der Lokalisierung von Proxies und Webcaches sowie der rein theoretischen Ansätze zur Lösung von Lokationsproblemen aus der mathematischen Programmierung und dem Operations Research. Weiterhin kann eine Klassifikation anhand der in den Arbeiten zu Grunde gelegten *Formen der Problembeschreibung* vorgenommen werden. Ausprägungen dieses Merkmals sind p-Median bzw. p-Zentrum Probleme, WLP/FLP, Set-Cover Probleme oder vollständig individuelle Ansätze. Als letztes Merkmal für die Unterscheidung der Arbeiten kann das zum Einsatz kommende *Lösungsverfahren* herangezogen werden. Hierbei lassen sich Arbeiten unterscheiden, welche auf Optimierungsverfahren (z. B. der linearen Programmierung) basieren oder auf den Einsatz von Heuristiken zurückgreifen.

Der weitere Verlauf dieses Abschnitts wird anhand des Anwendungsfelds der einzelnen Arbeiten untergliedert. Eine Übersicht der nachfolgend diskutierten Arbeiten sowie deren Vergleich auf Basis der eben beschriebenen Merkmale wird in Tabelle 13 wiedergegeben. Innerhalb der Tabelle werden die folgenden Abkürzungen verwendet (siehe auch Tabelle 12): die Anwendungsfelder der verschiedenen Lokalisierungsverfahren werden in Überwachungseinheiten (ÜE), Proxies und Caches (P+C) sowie Theorie (T) unterschieden. Die Problembeschreibung erfolgt als p-Median/p-Zentrum Probleme (pM), Set-Cover Probleme (SC), FLP sowie individuellen Ansätzen (IA). Bei den Lösungsverfahren wird in optimale Verfahren (O) und Heuristiken (H) unterschieden. Kombinationen mehrerer Ausprägungen werden durch „+“ verbunden dargestellt.

3.7.1 *Lokalisierung von Überwachungseinheiten*

Sämtliche der in diesem Abschnitt vorgestellten verwandten Arbeiten beziehen sich auf die Überwachung nicht-funktionaler Anforderungen an Netzwerke bzw. an einzelne Datenströme, ohne hierbei die jeweils zu überwachenden Dienstgütemerkmale genauer zu spezifizieren.

	Anwendungs-feld	**Problem-beschreibung**	**Lösungs-verfahren**
[SGKT05]	ÜE	SC+FLP	H+O
[CFL+05]	ÜE	SC	O
[LYCL05]	ÜE	SC+FLP	-
[CIB+06]	ÜE	IA	O
[HLO03]	ÜE	SC	-
[CS03]	P+C	SC	H
[Ono4]	P+C	IA	H+O
[QPV01]	P+C	pM	H
[Pla06]	T	pM+FLP	H
[MW05]	T	FLP	H
[WZZ06]	T	FLP	H
[XZ07]	T	FLP	H+O

Tabelle 13: Übersicht verwandter Arbeiten

Die erste verwandte Arbeit, die in diesem Abschnitt vorgestellt wird, stammt von Suh et al. (vgl. [SGKT05]). Die Autoren beschreiben unterschiedliche Ansätze zur Verteilung von Überwachungseinheiten sowie zur Bestimmung der optimalen Sampling-Raten der jeweiligen Überwachungseinheiten. Hierzu greifen die Autoren auf Verfahren zurück, die entweder eine Minimierung der gesamten Kosten oder die Maximierung der möglichen Abdeckung der zu überwachenden Infrastruktur als Zielsetzung haben. Der Schwerpunkt liegt dabei auf passiven Überwachungsansätzen mit und ohne Steuerung der Sampling-Raten. Zu den unterschiedlichen Verfahren werden mögliche Lösungsverfahren diskutiert, wobei der Fokus auf eigenentwickelten Heuristiken liegt. Insbesondere untersuchen die Autoren den Einfluss von zusätzlichen Überwachungseinheiten auf die Qualität der überwachten Lösung.

Von besonderem Interesse ist das als *Minimum Deployment and Operating Cost Problem without Sampling* bezeichnete Lokationsproblem, welches dieselben Zielsetzungen bei der Lösung wie das in dieser Arbeit beschriebene MULP aufweist. Die Problemdefinition erfolgt auf Grundlage eines unkapazitierten FLP, wobei allerdings kein Lösungsverfahren angegeben wird. Die Untersuchung geeigneter Heuristiken hierfür wird als Gegenstand zukünftiger Arbeiten genannt.

Einer vergleichbaren Fragestellung widmen sich auch Chaudet et al., die ebenfalls sowohl die Lokalisierung von passiven und zusätzlich aktiven Überwachungseinheiten als auch die Bestimmung der Sampling-Raten dieser Einheiten untersuchen (vgl. [CFL+05]). In allen Fällen kommen Verfahren der linearen Programmierung zum Einsatz. Die Autoren beschreiben den passiven Fall in Form eines speziellen Abdeckungsproblems, welches nicht die vollständige Überwachung einer Topologie sicherstellt. Das so genannte *Partial Passive Monitoring* Problem, welches ein Minimum-Set-Cover Problem darstellt, wird zur effizienten Lösung in ein die Kosten der Kommunikation über die Kanten der Infrastruktur minimierendes Problem

überführt. Als Lösungsverfahren kommen dabei Verfahren für ganzzahlige oder gemischt-ganzzahlige lineare Programme zum Einsatz.

Der wichtigste Beitrag der Arbeit von Chaudet et al. ist allerdings die konzeptionelle Herleitung und Begründung der jeweiligen Problemformulierungen, die in vergleichbarer Form auch in dieser Arbeit Anwendung finden. Die von den Autoren angebotenen Lösungsverfahren erfüllen hingegen nicht die in Abschnitt 3.5.3 diskutierten Anforderungen.

Ausschließlich auf die Platzierung aktiver Überwachungseinheiten innerhalb eines Netzwerks fokussieren sich Liu et al. in ihrer Arbeit (siehe [LYCL05]). Die Autoren untersuchen die Lokalisierung von Überwachungseinheiten bei gleichzeitiger Berücksichtigung der Maximierung der Abdeckung von Datenströmen sowie der Minimierung der gesamten Überwachungskosten. Die Problemformulierung erfolgt zunächst als allgemeines Abdeckungsproblem (All-Edges-Cover) unter der Berücksichtigung von Betriebs- und Installationskosten, welches nachfolgend in ein unkapazitiertes FLP überführt wird. Die Modellierung als FLP ist mit der Modellierung des in dieser Arbeit beschriebenen MULP vergleichbar, auch wenn sich die Zielfunktionen unterscheiden.

Es erfolgt keine Evaluierung der von Liu et al. vorgestellten Lösungsverfahren. Der Beitrag der Autoren liegt auch hier in der Spezifikation von Problemen und in den Hinweisen zu deren optimalen Lösung, weniger in der Lösung selbst.

Der Beitrag von Cantieni et al. befasst sich ebenfalls mit der Untersuchung der Lokalisierung von Überwachungseinheiten sowie der Ermittlung der notwendigen Sampling-Raten (vgl. [CIB+06]). Der vorgestellte Ansatz ist, im Vergleich zu den vorangegangenen Ansätzen, prototypisch implementiert und greift zur Bestimmung der Lokalisierung als auch zur eigentlichen Überwachung auf NetFlow-fähige[17] Netzwerkkomponenten zurück. Die Zielsetzung des beschriebenen Lokalisierungsansatzes ist die Maximierung des Nutzens, welcher aus der Überwachung des Datenstroms unter einer Platzierung sowie einer gegebenen Sampling-Rate entsteht. Zur Lösung des Maximierungsproblems wird die Formulierung in eine Lagrange-Form überführt und anschließend durch ein Gradientenverfahren gelöst.

Im Unterschied zu den zuvor beschriebenen Arbeiten und auch dem in dieser Arbeit entwickelten Verfahren befassen sich die Autoren intensiv mit der Fragestellung, inwiefern durch Verbindungsausfälle in realen Netzwerken die getroffenen Lokalisierungsentscheidungen beeinflusst werden. Zur Evaluation ihres Lösungsansatzes sowie der Untersuchung der eben genannten Fragestellung führen die Autoren umfangreiche Experimente auf Basis realer Messdaten aus dem Forschungsnetzwerk GEANT[18] durch.

Die letzte Arbeit, die in dieser Kategorie vorgestellt wird, stammt von Horton und López-Ortiz und beschäftigt sich mit der Bestimmung der Anzahl von Überwachungseinheiten innerhalb eines Netzwerks, welches das Border Gateway Protocol (BGP) zum Routing verwendet (vgl. [HLO03]). Zielsetzung des Ansatzes von Hor-

17 NetFlow ist eine von Cisco entwickelte Geräteschnittstelle für Messinformationen von Netzwerkkomponenten, welche in der Praxis eine hohe Verbreitung besitzt.

18 GEANT ist ein pan-europäisches Datennetzwerk für Forschungszwecke – vgl. http://www.dante.net/ – abgerufen am 14.04.2009.

ton und López-Ortiz ist die Platzierung von Instrumentierung zum Überwachen der Konnektivität eines Netzwerks. Hierbei soll die Anzahl der zu platzierenden Einheiten bei Kenntnis der Topologie sowie dem verwendeten Routingverfahren so minimiert werden, dass für jede Verbindung zweier Knoten innerhalb eines Netzwerks zu jedem Zeitpunkt festgestellt werden kann, ob die Verbindung noch besteht. Auch in dieser Arbeit kommt eine Minimum-Set-Cover Formulierung des Problems zur Anwendung. Im Rahmen der Arbeit wird keine Evaluation auf Basis von Messung vorgenommen, sondern die Anwendbarkeit des Verfahrens bewiesen.

Die Arbeit von Horton und López-Ortiz ist auf Grund der Übereinstimmung ihrer Zielsetzung mit einem der Nebenziele der vorliegenden Arbeit mit dieser verwandt. Die Autoren liefern Grenzen für die Anzahl der zu platzierenden Überwachungseinheiten bei einer gegebenen Topologie, wobei sie weitere Parameter eines Szenarios, wie z. B. die Anzahl der beteiligten Dienstnutzer, nicht berücksichtigen.

3.7.2 *Lokalisierung von Proxies und Webcaches*

Die erste Arbeit aus dem Anwendungsbereich der Lokalisierung von Proxies und Webcaches stammt von Choi und Shavitt und beschäftigt sich schwerpunktmäßig mit der Platzierung von transparenten und nicht-transparenten Servern in Netzwerken (siehe [CS03]). Transparente Server kennzeichnen hierbei jegliche Form von Servern, deren Existenz innerhalb einer Infrastruktur den Nutzern von Diensten und Informationsquellen verborgen bleiben soll. Hierunter fallen z. B. Webcaches oder aber Content-Stores in Content Distribution Networks (CDN). Aus Sicht dieser Arbeit wird nachfolgend nur auf den Fall der transparenten Server eingegangen, welcher aus inhaltlicher Sicht Parallelen zu der in dieser Arbeit untersuchten Fragestellung aufweist.

Die Autoren definieren das Lokationsproblem für transparente Server als Set-Cover Problem, wobei anstelle der Maximierung der Kantenabdeckung innerhalb einer Topologie die Maximierung der Abdeckung von Sessions, d. h. Routenbündeln, Zielsetzung ist. Zur Lösung des Problems werden einfache Heuristiken entwickelt und deren Anwendbarkeit auf Basis von Simulationsexperimenten untersucht.

Eine weitere Arbeit in diesem Anwendungsfeld, die sich mit der Platzierung von hochgradig verteilten Content-Store-Repliken beschäftigt, ist die Arbeit von On (vgl. [On04]). Am Beispiel von CDN sowie P2P-Netzen werden statische und dynamische Platzierungsstrategien zur Verbesserung der Gesamtverfügbarkeit eines Systems untersucht. Hierzu wird das Konzept der *Quality of Availability* eingeführt, welches die Verfügbarkeit eines Systems als wichtigstes zu steuerndes Merkmal definiert. Der Autor analysiert in der Arbeit im Rahmen von Simulationsexperimenten zufällig generierte Topologien im Hinblick auf die notwendige Anzahl von Content-Stores, den Ort ihrer Platzierung sowie der Granularität von Content-Store-Repliken. Zur Lösung der Platzierungsfragestellung kommen sowohl exakte Verfahren als auch Heuristiken zum Einsatz.

Besondere Merkmale der Arbeit von On sind der Einsatz kooperativer Heuristiken für die Platzierung von Content-Stores innerhalb von P2P-Netzen sowie die Verwendung eines Zugangskontrollansatzes in ressourcenbeschränkten Szenarien.

Qiu et al. adressieren in ihrer Arbeit ebenfalls die Platzierung von Webserver-Repliken in CDN, allerdings in Abgrenzung zu den vorangegangenen Beiträgen unter Einbezug des Nutzerverhaltens (vgl. [QPV01]). Die Einbindung des Verhaltens eines Nutzers von Webinhalten stellt die Parallele zu dem in dieser Arbeit beschriebenen Ansatz dar, welcher gleichermaßen auf Wissen über den Nutzer von Korrekturleistungen zurückgreift.

Die Autoren setzen zur Modellierung des Problems ein unkapazitiertes Minimum-p-Median Problem ein, untersuchen zuvor allerdings auch die Anwendbarkeit des FLP für das gegebene Szenario. Zur Lösung des Problems kommt eine Reihe von Heuristiken zum Einsatz. Im Speziellen werden eine quasi-optimale Lösung auf Basis einer Lagrange-Relaxation, baumbasierte Heuristiken, rein zufällige Platzierungen von Webserver-Repliken, sowie die Platzierung in der Nähe der Nutzer mit den höchsten Bedarfen (Hot Spot) untersucht und miteinander verglichen. Darüber hinaus entwickeln die Autoren einen gierigen Lösungsansatz, der das geschätzte Nutzerverhalten mit einbezieht. Die Untersuchung, die auf der Simulation der Verfahren mit zufälligen als auch realen Topologien basiert, hat zum Ergebnis, dass das Hot Spot Verfahren sowie der gierige Ansatz der Autoren die besten Ergebnisse liefert.

Hervorzuheben ist in der Arbeit von Qiu et al. die Berücksichtigung von unvollständigen Informationen, welche in der Simulation durch eine Verunreinigung der bekannten Daten durch Rauschen erreicht wird. Es zeigt sich, dass die untersuchten Verfahren allesamt robust gegenüber Störungen sind. Weiterhin zeigt sich, dass der Performanzgewinn durch Anwendung von Verteilungsverfahren unabhängig von der Topologie positiv ausfällt.

3.7.3 *Theoretische Ansätze zur Lösung von Lokationsproblemen*

Dieser Abschnitt beschäftigt sich mit theoretischen Arbeiten, welche mit der Problemstellung dieser Arbeit verwandt sind oder diese geeignet ergänzen. Auch wenn die vorgestellten Verfahren sehr gute Verteilungen liefern, so lassen sie sich auf Grund der Nichterfüllung der in Abschnitt 3.5.3 diskutierten Anforderungen nicht direkt für diese Arbeit nutzen.

Für eine weitergehende Diskussion theoretischer Lösungsansätze für klassische FLP bzw. WLP wird an dieser Stelle auf die einschlägige Operations Research Literatur verwiesen (siehe z. B. [DK97], [Vaz01] und [STA97]).

In der Arbeit von Plaxton (vgl. [Pla06]) werden Verfahren zur Lösung hierarchischer p-Median und FLP entwickelt und im Hinblick auf das Laufzeitverhalten sowie die Lösungsgüte untersucht. Plaxton definiert ein hierarchisches FLP, zu dessen Lösung inkrementelle Ansätze zur Lösung von FLP schrittweise in Richtung eines hierarchischen Ansatzes erweitert werden. Hierarchische Modelle sind eine Form der mehrstufigen FLP, welche Entscheidungen auf unterschiedlichen Ebenen abbilden können. Plaxton schlägt zur Lösung verschiedene Algorithmen vor, deren Existenz er beweist und deren Laufzeit und Lösungsgüte analytisch abgeschätzt werden.

Der Ansatz von Plaxton kann die Basis für eine Erweiterung des MULP sowie der in dieser Arbeit vorgestellten Lösungsansätze darstellen. Sobald die MAU nicht ausschließlich die Kontrolle an die aufrufende Instanz zurückgeben, sondern an an-

dere MAU delegieren können, ist ein mehrstufiger hierarchischer Ansatz notwendig.

Einen verteilten Ansatz zur Lösung des FLP beschreiben Moscibroda und Wattendörfer (vgl. [MW05]). Auf Grund der Überlegung, dass innerhalb einer Vielzahl von Szenarien keine globale Sicht auf die Infrastruktur und die benötigten Informationen möglich ist, definieren die Autoren einen verteilten Lösungsansatz für die Approximation an eine optimale Lösung eines unkapazitierten FLP. Dabei werden Teile des Lösungsverfahrens sowohl auf Seiten der Nutzer als auch der Facilities selbst zur Ausführung gebracht. Bei der Umsetzung kommt eine Form der primaldualen Approximation linearer Programme zum Einsatz. Bei der Lösung spielen die Berücksichtigung der durch eine Verteilung zusätzlich entstehenden Kommunikationskosten und deren Verhältnis zu den entstehenden Kostenersparnissen eine wichtige Rolle.

Xu und Zhang beschreiben in [XZ07] einen Spezialfall eines unkapazitierten FLP, welcher neben der Lokalisierung geeigneter Standorte für Facilities und der Zuordnung von Nutzern zu diesem Standorten ebenfalls noch unterschiedliche Dienstleistungen unterscheidet, welche durch eine Facility auf einem Standort erbracht werden können. Das von Xu und Zhang vorgestellte Verfahren verwendet eine LP Relaxation sowie stochastisch initiiertes Errichten und Schließen von Facilities. Die Autoren führen den Beweis, dass für die Annahme der Unabhängigkeit der Installationskosten vom Ort der Platzierung eine sehr hohe Approximationsgüte an eine optimale Lösung erreicht werden kann.

Die Bedeutung dieser Problemformulierung sowie des damit verbundenen Lösungsverfahrens für das Ausgangsszenario dieser Arbeit wird nachfolgend zusammen mit der nächsten verwandten Arbeit diskutiert.

Abschließend wird ein weiterer Spezialfall vorgestellt, welcher aus Sicht dieser Arbeit interessant ist. Eine Erweiterung des kapazitierten FLP betrachten Wu et al. (vgl. [WZZ06]). Die Autoren untersuchen Lösungsansätze für ein FLP, in welchen verschiedene Facilities auf einem Knoten gleichzeitig mit unterschiedlichen Installationskosten eröffnet werden können. Die Installationskosten hängen dabei von der Größe der zu platzierenden Facility ab. Die Modellierung erfolgt als gemischt-ganzzahliges lineares Programm, welches anschließend durch eine Lagrange-Relaxation gelöst wird. Weiterhin wird eine passende Lagrange-Heuristik entwickelt, welche Grenzen für das Optimierungsproblem auf Basis der Lagrange-Relaxation ermittelt. Die Güte der Heuristik wird anhand von zufällig generierten Daten als auch realen Daten in Experimenten untersucht und für große Modelle bestätigt.

Im Rahmen dieser Arbeit sind die Kosten einer Platzierung auf einem Knoten ausschließlich durch den Knoten selbst definiert und für jede zu installierende MAU gleich. Eine Unterscheidung der Installationskosten auf Basis des Typs oder der Herkunft einer MAU, wie diese in den beiden letzten Arbeiten beschrieben wird, ist innerhalb eines realen Szenarios in jedem Fall sinnvoll. Aus Sicht der Praxis werden mögliche Platzierungen von MAU und die damit verbundenen Kosten zwischen den beiden involvierten Parteien ausgehandelt und in Verträgen spezifiziert.

3.8 ZUSAMMENFASSUNG

Das dritte Kapitel dieser Arbeit untersucht Strategien zur Verteilung von Überwachungs- und Steuerungseinheiten innerhalb einer dienstbasierten Infrastruktur. Als Erweiterung des bisherigen Stands der Forschung werden verwandte Ansätze zur Verteilung von Überwachungseinheiten in Netzen um die Berücksichtigung notwendiger Steuerungs- bzw. Korrekturmaßnahmen bei der Ermittlung einer Verteilungsentscheidung ergänzt. Darüber hinaus finden erstmals Merkmale dienstbasierter Infrastrukturen, wie beispielsweise die Abbildung der Nutzung fremder Kontrollsphären während eines Dienstaufrufs, Eingang in die zur Anwendung kommenden Lösungsverfahren.

Nach Diskussion der Anforderungen an eine Verteilung von Überwachungs- und Steuerungseinheiten wird diese als mathematisches Optimierungsproblem formuliert, welches als Monitoring Unit Location Problem bezeichnet wird. Das Optimierungsproblem wird unter Verwendung gemischt-ganzzahliger Optimierungsverfahren aus dem Bereich des Operations Research zur Identifikation der optimalen Platzierung gelöst. Da das dem MULP zu Grunde liegende WLP als NP-schwer bekannt ist, werden darüber hinaus geeignete Näherungsverfahren entwickelt, welche ein besseres Laufzeitverhalten sowie eine geringere Komplexität als das optimale Verfahren aufweisen. Im weiteren Verlauf werden zwei Heuristiken definiert und evaluiert, welche das Optimierungsproblem in linearer an Stelle von exponentieller Laufzeit mit einer ausreichend hohen Güte lösen.

Die Auswertung der Ergebnisse aus der Untersuchung von Topologien mit 100 Knoten, welche in diesem Kapitel als Referenz herangezogen werden, zeigt, dass im schlechtesten Falle durch das Verbesserungsverfahren immer noch mehr als 85 % der Lösungsgüte des optimalen Ansatzes erreicht werden. Diese Lösungsgüte wird in nur einem Viertel der Laufzeit der optimalen Lösung errechnet. Im besten Falle hingegen lässt sich durch das Verbesserungsverfahren eine Lösungsgüte von 99,16 % bei einer relativen Laufzeit von 8,77 % erreichen. Die Untersuchung der Einsparpotentiale liefert ebenfalls positive Indikatoren der Nützlichkeit einer Verteilung: während im schlechtesten Falle durch das Verbesserungsverfahren lediglich eine Einsparung von 17,21 % erreicht wird, so ermöglicht die Verteilung im besten Falle eine Ersparnis von über 54 % der Kosten des zentralen Ansatzes. Die Anzahl der benötigten Überwachungseinheiten schwankt zwischen 22 % im besten Falle bis hin zu 70 % der Dienstanbieter im schlechtesten Falle.

Die Ergebnisse der Untersuchungen zeigen, dass in allen untersuchten Fällen Kosteneinsparungen durch die Verteilung von Überwachungs- und Steuerungseinheiten möglich sind und das die Heuristiken, im Besonderen das auf dem Simulated Annealing Ansatz basierende Verbesserungsverfahren, sehr gut dazu geeignet sind, eine Verteilung von Einheiten innerhalb einer Infrastruktur bei einem akzeptablen Laufzeitverhalten zu ermitteln.

4

EINE SPEZIFIKATIONSSPRACHE FÜR ANFORDERUNGEN UND REAKTIONEN AUF ABWEICHUNGEN

In diesem Kapitel wird eine leichtgewichtige Spezifikationssprache entwickelt, mit deren Hilfe gleichermaßen Anforderungen an einen Dienst als auch Reaktionen auf Abweichungen von diesen Anforderungen spezifiziert werden können. Die mit *Web Service Requirements and Reactions Policy* bezeichnete Sprache erlaubt eine solche Beschreibung für Webservice-basierte Dienste (vgl. hierzu [RMNS08] und [RES+08]). Als Bindeglied zwischen fachlichen Anforderungen an Dienste und der Umsetzung einer verteilten Überwachung und Steuerung beschreibt die auch als WS-Re2Policy bezeichnete Sprache die Handlungsspielräume der Überwachungs- und Steuerungseinheiten in einer dienstbasierten Infrastruktur.

Das weitere Kapitel ist wie folgt gegliedert: Zuerst werden die Problemstellung sowie die in diesem Kapitel untersuchten Forschungsfragestellungen diskutiert. Insbesondere wird erläutert, warum ein Bedarf für eine neue Spezifikationssprache im Zusammenhang mit dem in dieser Arbeit zu Grunde gelegten Szenario besteht. Danach werden die Anforderungen an eine solche Sprache definiert, bevor im folgenden Abschnitt die Sprache und ihre wichtigsten Bestandteile beschrieben werden. Die Vorstellung verwandter Arbeiten sowie eine Zusammenfassung bilden den Abschluss dieses Kapitels.

4.1 PROBLEMSTELLUNG

In dieser Arbeit wurde bereits ausführlich erläutert, dass die tatsächlichen Leistungsmerkmale eines Dienstes nicht immer den zugesicherten oder erwarteten Leistungsmerkmalen entsprechen müssen. Im Falle von Abweichungen müssen geeignete Maßnahmen initiiert werden. Bisherige Ansätze, wie beispielsweise das zuvor vorgestellte WSLA, sehen grundsätzlich die Notwendigkeit der Definition von Reaktionen auf Abweichungen, wobei sie diese aber nicht weiter spezifizieren. Gerade eine Verteilung der Überwachung und Steuerung ist allerdings nicht oder nur stark eingeschränkt möglich, wenn die Kontrolle zur Behandlung von Abweichungen von einer verteilten Überwachungseinheit an eine zentrale Steuerungsinstanz zurück übergeben werden muss, da nur diese mögliche Reaktionen auf die vorliegenden Abweichungen kennt. Aus Sicht dieser Arbeit ist es deshalb notwendig, dass Reaktionen auf Abweichungen parallel zu den Anforderungen spezifiziert werden können und den verteilten Überwachungs- und Steuerungseinheiten zur Verfügung stehen. Die vorliegende Arbeit untersucht, wie eine solche Spezifikationssprache gestaltet sein muss, um leichtgewichtig und einfach erweiterbar die gestellten Anforderungen erfüllen zu können.

Darüber hinaus ist die Spezifikation der möglichen und durch den Auftraggeber der Überwachung akzeptierten Gegenmaßnahmen von großer Bedeutung. Die zum Zeitpunkt der Entwicklung von WS-Re2Policy existierenden Ansätze sehen als ein-

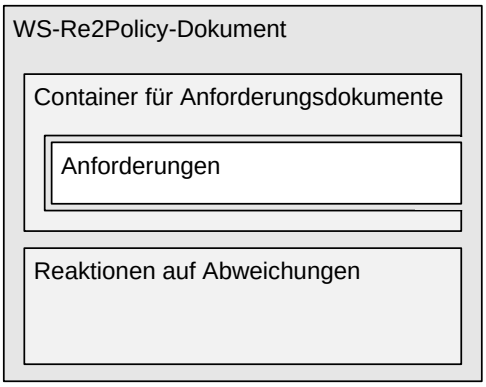


Abbildung 19: WS-Re2Policy als Container

zig zulässige Gegenmaßnahme die Meldung des Vorfalls an die aufrufende Instanz vor. Die im Bereich der dienstbasierten Workflows intensiv diskutierten Anpassungsmechanismen, wie z. B. die Selektion alternativer Dienste und das damit verbundene Neuplanen der Workflows, werden von ihnen nicht ausreichend unterstützt. In der vorliegenden Arbeit werden mögliche Gegenmaßnahmen untersucht und in die WS-Re2Policy-Sprache integriert. Aktuelle Sprachentwicklungen, wie beispielsweise die in Abschnitt 4.4 dargestellten verwandten Arbeiten, haben mittlerweile die Notwendigkeit entsprechender Erweiterungen erkannt und berücksichtigen diese im jeweiligen Sprachentwurf.

4.2 ANFORDERUNGEN AN EINE SPEZIFIKATIONSSPRACHE

Dieser Abschnitt beschreibt die Anforderungen an eine neue Spezifikationssprache für Anforderungen und Reaktionen auf Abweichungen von diesen Anforderungen. Hierbei wird eine Unterscheidung in anwendungsfallspezifische Anforderungen als auch in umsetzungsbezogene Anforderungen vorgenommen.

Die anwendungsfallspezifischen Anforderungen lassen sich aus dem geplanten Einsatzszenario der Spezifikationssprache ableiten. Ziel ist es, in kompakter Form für Überwachungs- und Steuerungseinheiten zu definieren, welche Anforderungen sie einerseits zu überwachen haben und welchen Handlungsspielraum sie andererseits bei Abweichungen nutzen können, um Abweichungen zu korrigieren und einen ordnungsgemäßen Systemzustand wiederherstellen zu können. Demnach muss die zu entwickelnde Sprache beide Komponenten gleichermaßen abbilden können. Wünschenswert ist darüber hinaus eine Nutzung der Sprache auch dann, wenn ausschließlich Anforderungen beschrieben werden sollen und eine Abweichungsbehandlung nicht erwünscht ist.

Darüber hinaus gibt es eine Reihe von Anforderungen aus Sicht einer praktischen Umsetzung einer solchen Spezifikationssprache. Aus technischer Sicht sollten zur Verarbeitung der Spezifikationssprache keine zusätzliche Softwarekomponenten benötigt werden, wie sie beispielsweise regelbasierte Ansätze bedingen und die gerade auf Geräten mit Ressourcenbeschränkungen unnötigen Overhead erzeugen. Weiterhin sollte eine zu entwickelnde Spezifikationssprache nicht vollkommen isoliert von bisherigen Anforderungsspezifikationssprachen existieren, sondern diese in geeigneter Weise integrieren. Im Webservice-Umfeld existiert eine Reihe von Sprachen

auf der Basis des WS-Policy-Frameworks, welches als W3C-Standard eine breite Akzeptanz besitzt. Eine Integration mit dieser Sprachfamilie ist somit aus Gründen der Akzeptanz sowie einer möglichen Erweiterbarkeit notwendig. Um eine gänzliche Neuentwicklung von Sprachelementen zur Spezifikation von Anforderungen (z. B. im Hinblick auf Dienstgütemerkmale) zu vermeiden, soll die zu entwickelnde Sprache einen Container für existierende Sprachelemente bilden und diese mit der Eigenschaft der Spezifikation von Reaktionen auf Abweichungen ergänzen (siehe Abbildung 19). Eine letzte Anforderung stellt die leichte Lesbarkeit der Spezifikationssprache durch menschliche Nutzer dar.

4.3 WEB SERVICE REQUIREMENTS AND REACTIONS POLICY

In diesem Abschnitt wird die WS-Re2Policy-Sprache im Detail vorgestellt. In Unterabschnitt 4.3.1 erfolgt die Diskussion der Grundlagen der zu entwickelnden Sprache, während in Unterabschnitt 4.3.2 die Kernelemente der Sprache sowie ihre Struktur auf Basis der XML vorgestellt wird. Den Abschluss bildet die Vorstellung eines Anwendungsbeispiels in Unterabschnitt 4.3.3.

4.3.1 *Grundlagen des Sprachentwurfs*

Der Entwurf der WS-Re2Policy-Sprache basiert auf zwei Kernkonzepten. Zunächst gilt es die Kompatibilität des Containers für Anforderungselemente zu der durch das W3C standardisierten WS-Policy-Sprache in Version 1.2 zu gewährleisten, um eine Akzeptanz der Sprache im Webservice-Umfeld zu erleichtern sowie die einfache Erweiterbarkeit der Sprache durch WS-Policy-Sprachelemente sicherzustellen. Darüber hinaus soll zur Beschreibung der Reaktionen auf Abweichungen auf einen etablierten Ansatz zur Spezifikation von Regeln zurückgegriffen werden, welcher einfach verständlich ist und gleichzeitig eine hohe Ausdruckskraft besitzt.

In einem ersten Schritt werden aus diesem Grund die Kernelemente der WS-Policy-Sprache beschrieben. Der WS-Policy-Standard kennt auf abstrakter Ebene zwei Grundelemente, aus welchen Policy-Dokumente auf XML-Basis zusammengesetzt werden [BBC+06]:

- *Zusicherungen* sind die Elemente innerhalb eines Dokuments, welche einzelne Anforderungen beinhalten.
- *Alternativen* fassen die Zusicherungen zusammen und bilden somit eine implizite Hierarchie zwischen den Zusicherungen ab. Hierzu kann sie kennzeichnen, ob nur eine einzelne Zusicherung (*ExactlyOne*) oder alle Zusicherungen (*All*) innerhalb einer Alternative ausgeführt werden müssen.[19]

Es gilt anzumerken, dass unter den Alternativen selbst keine Reihenfolge definiert werden kann. Ein System, welches die Policy verarbeiten soll, muss die der vorliegenden Policy zu Grunde liegende Semantik kennen. Alternativen können ineinander verschachtelt werden, um komplexere Strukturen abbilden zu können. Allerdings

19 Im Detail unterscheidet der Standard hier in eine kompakte Darstellungsform sowie eine Normalform, welche umfangreicher ist – vgl. hierzu [BBC+06].

kann eine Policy auch leer sein, d. h. sie muss nicht zwangsläufig eine Alternative zu einer Anforderung spezifizieren. Ebenfalls kann auch aus Sicht des Policy-Frameworks keine Unterscheidung zwischen den Inhalten der einzelnen Alternativen vorgenommen werden. Dieselben Zusicherungen können sich gleichzeitig in mehreren Alternativen befinden, ohne dass das aus Frameworksicht festgestellt werden kann.

Zahlreiche Erweiterungen und Ergänzungen zu WS-Policy, wie z. B. die Web Services Policy Assertions Language [BHK+03], definieren darüber hinaus ergänzende Semantik zur Auszeichnung der jeweiligen Zusicherungen. Durch ihren Einsatz wird beispielsweise spezifiziert, unter welchen Bedingungen eine Zusicherung angewendet werden muss (*in jedem Fall* oder *optional*) und welchen Einfluss diese auf die Weiterverarbeitung hat (*Fehlermeldung wird ausgelöst und Ausführung angehalten* oder *keine Behandlung von Fehlern*).

Das zweite Kernkonzept der WS-Re2Policy-Sprache adressiert die Beschreibung der Reaktionen auf Abweichungen von den durch WS-Policy spezifizierten Anforderungen. Theoretisches Fundament für die Modellierung der Reaktionen auf Abweichungen in WS-Re2Policy ist das ECA-Paradigma (*Event, Condition, Action*), welches implizit zur Beschreibung von Regelsätzen verwendet wird. ECA stammt aus dem Bereich aktiver Datenbanksysteme und findet im Umfeld sämtlicher ereignisgetriebener Systeme starke Anwendung (vgl. z. B. [WC95]).

Die Grundstruktur einer Modellierung auf Basis des ECA-Paradigmas lässt sich aus abstrakter Sicht wie folgt beschreiben:

```
ON Event
IF Condition
DO Actions
```

Konkret bedeutet diese Formulierung, dass ein Ereignis die Verarbeitung einer Regel anstößt, welche eine Bedingung überprüft und auf Basis der Evaluation der Bedingung eine Aktion durchführt. WS-Re2Policy bzw. dessen Bestandteile sowie die durch WS-Policy direkt modellierten Anforderungen lassen sich direkt auf dieses Paradigma übertragen und den jeweiligen Elementen von ECA wie folgt zuordnen:

- *Event*: Aktueller Messwert aus dem Überwachungssystem, beispielsweise die Antwortzeit eines Dienstes oder dienstbasierten Workflows.
- *Condition*: Zusicherungen, deren Einhaltung überprüft werden muss, z. B. ein oberer Grenzwert für die Antwortzeit eines Dienstes.
- *Actions*: Reaktionen, die ausgelöst werden, sobald eine Bedingung verletzt wird, wie beispielsweise der Neustart eines Dienstes nachdem dieser nicht geantwortet hat.

Die ECA-Elemente lassen sich an verschiedenen Stellen innerhalb der Spezifikationssprache wiederfinden, wenn auch nicht in der eben dargestellten klassischen Beschreibungsform *on event – if condition – do action*, sondern vielmehr in stark integrierter Form. Diese Integration wird im nachfolgenden Abschnitt erläutert.

Die Modellierung von Reaktionen auf Abweichungen mit einer einfachen regelbasierten Sprache bietet eine Reihe von Vorteilen. Zunächst einmal ist das ECA-Paradigma einfach in der Anwendung, d. h. auch Nicht-Experten können entsprechend modellierte Vereinbarungen lesen und anpassen. Gleichermaßen erlaubt der

Einsatz von ECA, ebenso wie der jeder anderen Regelsprache, eine klare Trennung der Kontrolllogik von der eigentlichen Implementierung des Systems, was die Wiederverwendung von Spezifikationsdokumenten ermöglicht.

4.3.2 *Kernelemente der Spezifikationssprache*

In diesem Abschnitt werden die wichtigsten Bestandteile der WS-Re2Policy-Sprache beschrieben. Besonderer Fokus liegt hierbei auf der Diskussion der möglichen Reaktionen auf Anforderungsabweichungen sowie deren Modellierung.

Wie im vorherigen Abschnitt bereits erwähnt, wird zur Umsetzung der WS-Re2-Policy-Sprache auf das WS-Policy-Framework des W3C in Version 1.2 zurückgegriffen. Dies ermöglicht die Verwendung sämtlicher zu WS-Policy konformen Sprachen innerhalb von WS-Re2Policy, so dass beispielsweise auch die WS-SecurityPolicy[20] zur Beschreibung von Sicherheitsanforderungen in die Spezifikationssprache eingebunden werden kann.

Analog zu der im Abschnitt 4.2 beschriebenen Zweiteilung einer Spezifikationssprache in Anforderungen und Reaktionen auf Abweichungen erfolgt die Strukturierung von WS-Re2Policy. Die beiden Hauptelemente werden demnach als *Requirements Part* und *Reactions Part* definiert. Der Anforderungsteil beinhaltet die Ereignisse, welche überwacht werden müssen, sowie die einzuhaltenden Bedingungen. Der Reaktionsteil beinhaltet demnach die auszuführenden Aktionen, wobei diese als Block zu sehen sind. Es wird pro Anforderungsteil, der aus den unterschiedlichsten Bedingungen bestehen kann, ein Reaktionsblock mit einem Steuerungsprogramm definiert. Alternative Steuerungsprogramme innerhalb einer Policy sind nicht zulässig.

Eine Darstellung aller Elemente der Sprache wird in Abbildung 20 wiedergegeben. Anforderungen können in jeder mit WS-Policy kompatiblen Form spezifiziert werden. Die Spezifikation der Merkmale stellt dabei nicht sicher, dass ein die Spezifikation verarbeitendes System die notwendigen semantischen Kenntnisse zu deren Verarbeitung besitzt. Eine vollständig in sich selbst beschriebene und damit abgeschlossene Definition von Anforderungen setzt den Einsatz semantischer Annotationen voraus, deren Betrachtung allerdings nicht Gegenstand dieser Arbeit ist.

In der aktuellen Version unterstützt WS-Re2Policy im *Requirements Part* verschiedene eigene Sprachelemente zur Beschreibung wichtiger Dienstgütemerkmale. Diese werden nativ durch die Sprache unterstützt, stellen aber nur eine Übergangslösung dar und sollen bei Erscheinen einer standardisierten QoS-Policy durch diese ersetzt werden. Die bisher abgebildeten Merkmale sind der Durchsatz, die Antwortzeit sowie die Verfügbarkeit eines Dienstes oder eines dienstbasierten Workflows.

Im weiteren Verlauf werden die einzelnen Elemente im Detail beschrieben. Neben der reinen Beschreibung von Anforderungen werden im *Requirements Part* zwei weitere Elemente definiert. Der komplexe Typ *RequirementsType*, welcher den *Requirements Part* definiert, beinhaltet das Attribut *Requirements ID* sowie *Requirements Needed*.[21] Das erste Attribut wird dazu verwendet, um eine Zuordnung von An-

20 Die WS-SecurityPolicy wird, im Gegensatz zu WS-Policy selbst, von OASIS standardisiert und liegt aktuell in Version 1.3 vor – vgl. [NGG+09].

21 Die Bezeichnung „komplexer Typ“ (engl. complex type) bezieht sich auf die Spezifikation von XML-Schemata und kennzeichnet einen zusammengesetzten Datentyp.

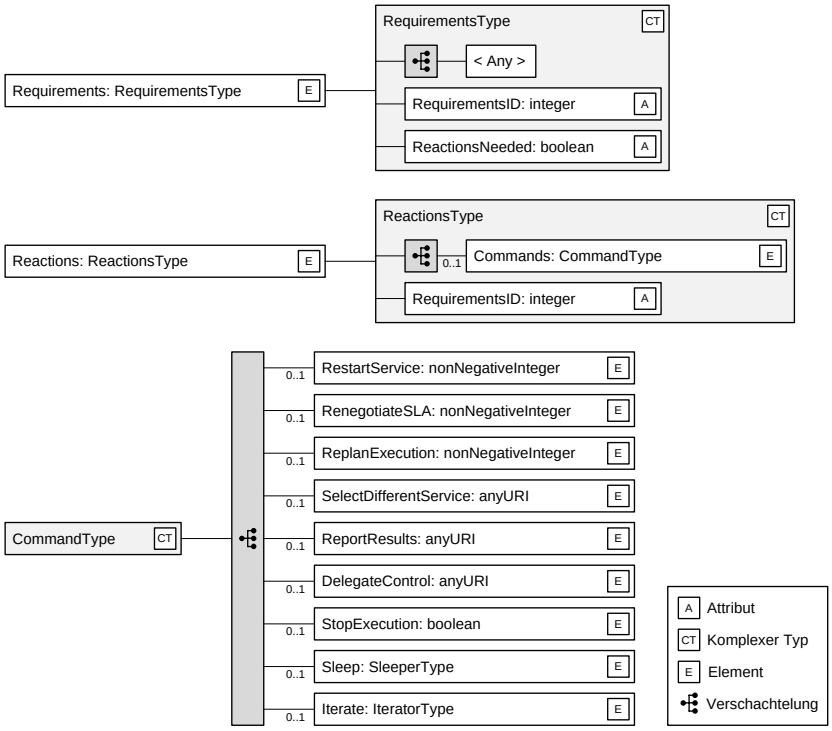


Abbildung 20: Sprachelemente der WS-Re2Policy-Sprache

forderungsdefinition zu *Reactions Part*, d. h. den zulässigen Gegenmaßnahmen auf Abweichungen, zu etablieren. Eine solche Zuordnungsmöglichkeit ist im Hinblick auf die Wiederverwendbarkeit einzelner Bestandteile eines Policy-Dokuments sinnvoll. Somit wird ermöglicht, dass ein *Reactions Part* in Verbindung mit verschiedenen *Requirements Parts* eingesetzt werden kann. Das zweite Attribut hingegen dient der Festlegung, ob überhaupt eine Behandlung von Abweichungen benötigt wird. Wie bereits zuvor erwähnt kann ein WS-Re2Policy-Dokument auch nur als Container für Anforderungen eingesetzt werden.

Der *Reactions Part*, welcher durch den komplexen Typ *ReactionsType* definiert, besitzt ebenfalls ein Attribut *Requirements ID* als Gegenstück zum zugehörigen Attribut im *Requirements Part*. Darüber hinaus beinhaltet das Element eine Sammlung von *Commands*, welche Gegenmaßnahmen sowie einfache Kontrollstrukturen abbilden. Diese Befehle können ihrerseits ineinander verschachtelt sein, um komplexere Kontroll- und Reaktionskonstrukte abzubilden.

Die nachfolgend aufgeführten Reaktionstypen werden aktuell durch die WS-Re2-Policy-Sprache unterstützt und ebenfalls im AMAS.KOM Framework abgebildet:

- *RestartService*: Der Befehl ermöglicht den Neustart eines Dienstes nachdem dieser eine Anforderung verletzt hat.
- *RenegotiateSLA*: Die Neuverhandlung von Anforderungsparametern, die wiederholt verletzt wurden, ist durch Anwendung dieses Befehls möglich.
- *ReplanExecution*: Das Kommando initiiert die Neuplanung eines vollständigen Ablaufplans, sofern die Einheit einen Workflow überwacht und steuert.

- *SelectDifferentService*: Die Auswahl eines alternativen Dienstes, welcher durch diesen Befehl gesteuert wird, ist das Gegenstück zur Neuplanung eines Workflows für nur einen Dienst.
- *ReportResults*: Der Befehl bildet die Möglichkeit ab, Ergebnisse einer Überwachung zur weiteren Verarbeitung an die aufrufende Instanz zu melden, wobei die Kontrolle nicht übergeben wird.
- *DelegateControl*: Die Delegation der gesamten Kontrolle an eine andere Überwachungs- und Steuerungseinheit oder aber an die aufrufende Instanz ist durch diesen Befehl modellierbar.
- *StopExecution*: Das Kommando dient dem kontrollierten Abbruch der Dienstnutzung und der Signalisierung eines Fehlers an die aufrufende Instanz.

Darüber hinaus werden verschiedene Kontrollkonstrukte angeboten, mit deren Hilfe einfache Überwachungs- und Steuerungsabläufe modelliert werden können. Der Befehl *Iterate* ermöglicht die Abbildung von Schleifen, die durch einen vorgegebenen Zähler oder auch eine zeitliche Begrenzung gesteuert werden. Der Befehl *Sleep* beschreibt eine zeitlich definierte Unterbrechung innerhalb eines Überwachungs- und Steuerungsablaufs.

4.3.3 *WS-Re2Policy anhand eines Beispiels*

In diesem Abschnitt wird die WS-Re2Policy-Sprache anhand eines Beispiels erläutert. Das nachfolgende Listing stellt ein XML-Dokument dar, welches zum WS-Re2Policy-Schema (siehe Abschnitt A.5 im Anhang dieser Arbeit) konform und mit Beispieldaten gefüllt ist. Aus Gründen der Lesbarkeit wurden die Namespace-Deklarationen von WS-SecurityPolicy und WS-Re2Policy entfernt.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Created by Nicolas Repp (KOM TUD) -->
<re2:RequirementsReactionsSuite ... >
 <re2:Requirements ReactionsNeeded="true" RequirementsID="3428">
  <wsp:Policy wsu:Id="ID_236" Name="Security-QoS">
   <wsp:All>
    <wsp:All>
     <sp:EncryptedParts>
      <sp:Body/>
     </sp:EncryptedParts>
    </wsp:All>
    <wsp:All>
     <qos:Throughput>10</qos:Throughput>
     <qos:ResponseTime>23.55ms</qos:ResponseTime>
    </wsp:All>
   </wsp:All>
  </wsp:Policy>
 </re2:Requirements>
 <re2:Reactions RequirementsID="3428">
  <re2:Sleep time="10.0ms"/>
  <re2:Iterate time="0.0ms" count="2">
```

```
   <re2:RestartService/>
  </re2:Iterate>
  <re2:DelegateControl>caller</re2:DelegateControl>
 </re2:Reactions>
</re2:RequirementsReactionsSuite>
```

Wenn man den Anforderungsteil des Dokuments im vorangegangenen Listing betrachtet, so kann man feststellen, dass zur Spezifikation von Anforderungen zwei auf WS-Policy basierende Sprachen zum Einsatz kommen. Die erste beschriebene Alternative beinhaltet eine Anforderung auf Basis der WS-SecurityPolicy, die ausdrückt, dass der Nachrichtenkörper einer Dienstinteraktion in jedem Fall verschlüsselt sein muss (definiert in den Elementen *EncryptedParts* und *Body*). Die zweite im Beispiel aufgeführte Alternative besteht aus zwei Zusicherungen, welche die nativ durch WS-Re2Policy unterstützten Dienstgütemerkmale abbilden (Elemente *Throughput* und *ResponseTime*). Die Dienstgüteparameter definieren eine Minimalanforderung an den Durchsatz des überwachten Dienstes von 10 parallelen Dienstaufrufen sowie eine dafür maximal zulässige Antwortzeit von 23,55 ms.

Wie bereits in Abschnitt 4.3.1 erwähnt, definiert WS-Re2Policy keine Semantik der einzelnen Elemente. Das verarbeitende System muss demnach selbst wissen, wie die in der Policy beschriebenen Anforderungen zu erfüllen sind. Die Ausführung des Reaktionsteils wird nur bei dem System bekannten Anforderungsteilen durchgeführt.

Die Gegenmaßnahmen auf Abweichungen sind im zweiten Abschnitt des Dokuments aufgeführt, wobei beide Teile des Policy-Dokuments über die *RequirementsID* „3428“ zusammengefügt werden. Im Beispiel aus dem vorangegangenen Listing wird der Dienst bei Verletzung einer der beiden Dienstgüteanforderungen zunächst einmal neu gestartet, nachdem ein Timer von 10 ms abgelaufen ist. Der Neustart, d. h. der erneute Aufruf, eines Dienstes stellte sich in praktischen Versuchsaufbauten häufig als effiziente Methode dar, eine Antwort von einem Dienst zu erhalten (vgl. [RBHS07]). Sollte nach zweimaligem Neustart des Dienstes immer noch keine Antwort vom Dienst feststellbar sein, so wird die Ausführung beendet und die Kontrolle an die aufrufende Instanz zurückgegeben.

4.4 VERWANDTE ARBEITEN

In diesem Abschnitt werden zu WS-Re2Policy verwandte Arbeiten vorgestellt und von den Inhalten dieser Arbeit abgegrenzt. Die Reihenfolge der vorgestellten verwandten Arbeiten orientiert sich an dem in den Arbeiten vorgestellten Funktionsumfang. Den Anfang bilden Arbeiten, in welchen ausschließlich Anforderungen definiert werden, die aber auf Grund des Anwendungsszenarios oder der verwendeten Technologien dem Ansatz dieser Arbeit verwandt sind.

Als Erstes werden an dieser Stelle zwei Arbeiten von Baresi et al. vorgestellt. Die Autoren beschreiben in [BG05] einen Ansatz, in welchem Anforderungen zur Überwachung als Teil des BPEL-Codes integriert werden. Durch Verwendung von Vor- und Nachbedingungen innerhalb der Kommentarfelder des Ausführungsplans können primär funktionale Anforderungen definiert werden. Eine Erweiterung stellt in

[BGP06] die Einführung der *Web Service Constraint Language* dar, welche auf Basis von WS-Policy eine Beschreibung der Anforderungen von Dienstnutzern, Dienstanbietern und dritten Parteien zulässt.

Der erweiterte Ansatz lässt sich insofern mit WS-Re2Policy vergleichen, da er ebenfalls WS-Policy verwendet und darüber hinaus eine am Aufruf eines Dienstes orientierte Bindung an einen Workflow besitzt. Die Arbeiten berücksichtigen keine Unterstützung der Behandlung von Abweichungen.

Eine weitere verwandte Arbeit stammt von Tosic et al., welche mit der *Web Service Offerings Language* (vgl. [TPP02], [TPP03] und [TME+06]) eine Spezifikationssprache definieren, die analog zu WS-Re2Policy im Bereich des Dienstgütemanagements Einsatz findet. Auch sie integriert sich anhand der Aufrufe, die durch einen Dienst innerhalb eines dienstbasierten Workflows durchgeführt werden. Die Sprache ermöglicht die Spezifikation von Anforderungen im Hinblick auf Dienstgütemerkmale aus Sicht der betroffenen Parteien und unterstützt bereits einfache Reaktionen auf Abweichungen, wobei die Autoren als Reaktionen auf Abweichungen monetäre Kompensationsmechanismen vorsehen. Komplexe Steuerungsstrukturen sind innerhalb der Sprache nicht vorgesehen.

Ein Ansatz, welcher ebenfalls einfache Reaktionen auf Abweichungen zur Durchsetzung von Politiken vorsieht, ist Bestandteil der von Ludwig et al. in [LDK04] beschriebenen CREMONA-Architektur. Analog zu WS-Re2Policy benutzen die Autoren WS-Policy als Bestandteil der WS-Agreement-Sprache zur Spezifikation von Anforderungen an Webservices. Allerdings liegt der Fokus der Arbeit auf der erstmaligen Verhandlung und späteren Nachverhandlung von gemeinsamen Anforderungen der Partnern und der Gestaltung der dabei durchzuführenden Interaktionen.

Von denselben Autoren stammt die im wissenschaftlichen Umfeld weit verbreitete WSLA-Sprache (siehe auch Abschnitt 2.3.3), welche als Bestandteil von Dienstgütevereinbarungen ebenfalls die Definition von Reaktionen auf SLA-Verletzungen vorsieht (vgl. z. B. [LKD+03a] und [LKD+03b]). Eine detaillierte Spezifikation der vorgesehenen Reaktionen ist allerdings explizit aus der umfangreichen Sprachbeschreibung ausgeklammert. Lediglich das Reporting von Abweichungen wird durch die Sprachbeschreibung vorgesehen.

Weiteres Abgrenzungsmerkmal ist der unterschiedliche Fokus bei der Spezifikation. Beide Arbeiten wurden zur Spezifikation vollständiger Geschäftsbeziehungen entwickelt und bilden die Sichten aller an einer Interaktion beteiligten Parteien ab, was der in dieser Arbeit geforderten Leichtgewichtigkeit widerspricht. WS-Re2Policy hingegen dient der Modellierung der Anforderungen und Reaktionen auf Abweichungen aus Sicht des Dienstnutzers, welcher durch eine verteilte Überwachung- und Steuerung seine Interessen schützen möchte.

Im Folgenden werden zwei Ansätze vorgestellt, die speziell für die Beschreibung von Anforderungen an vollständige Workflows entwickelt wurden. Die Spezifikation von Dienstgüteanforderungen hat die *Quality of Service Language for Business Processes* von Baligand et al. zum Ziel [BRL07]. In Ergänzung zur der vorgestellten Spezifikationssprache wird von den Autoren zusätzlich eine Erweiterung der Laufzeitumgebung für die durch die Sprache annotierten Workflows vorgeschlagen. Als Parallele zu WS-Re2Policy gilt es die starke Unterstützung von Gegenmaßnahmen

bei Anforderungsabweichungen zu nennen. Die Arbeit von Baligand et al. schlägt zu diesem Zweck bereits eine Reihe von Aktionen vor, wie z. B. das erneute Planen einer Ausführung, die Auswahl eines alternativen Dienstes oder aber die Neuverhandlung von Dienstgütemerkmalen.

Mit der von Comes et al. mehr als ein Jahr nach der erstmaligen Veröffentlichung von WS-Re2Policy vorgeschlagenen Sprache lassen sich gleichermaßen Anforderungen an einen vollständigen Workflow spezifizieren. Die Autoren verwenden ein auf Business Rules basierendes Format zur Beschreibung von Anforderungen und Reaktionen auf Abweichungen [CBZ09]. Erklärtes Ziel der BPRules-Sprache ist die Annotation von Workflows in einer Form, die sich auch auf die aus der Dekomposition eines Workflows entstehenden Bestandteile übertragen lässt und dabei die Besonderheiten einer Orchestration weiterhin erhält. Regelsätze können darüber hinaus zur Laufzeit angepasst werden.

Der Funktionsumfang der Sprache in Bezug auf die Modellierung von Reaktionen auf Abweichungen ist dem der WS-Re2Policy vergleichbar. Die Autoren unterscheiden hierbei in „Management Actions“, welche den Reaktionen in WS-Re2Policy entsprechen und zur Steuerung von Diensten dienen, sowie „General Actions“, die zum Management der Regelsätze selbst dienen (z. B. das Nachladen von Regelsätzen).

Zum Abschluss soll eine Spezifikationssprache aus dem Bereich der industriellen Anwendung vorgestellt werden. Die *Simplified Policy Language* ist eine auf die Anwendung im Umfeld des Common Information Model (CIM), einem Industriestandard zur Beschreibung von Managementdaten für unterschiedlichste Ressourcen und Dienste, zugeschnittene Sprache, die von der Zielsetzung her zu WS-Re2Policy verwandt ist. Kern der Policy-Sprache ist die Definition von Regeln für das Management der durch das CIM definierten Ressourcen [ACLL07]. Die Unterstützung von Reaktionen auf Abweichungen ist fester Bestandteil der Sprache. Auf Basis der im Informationsmodell beschriebenen Ressourcen und den durch diese zur Verfügung gestellten Daten können beliebige Regelsätze aufgebaut werden, die zur Steuerung der CIM-Ressourcen einsetzbar sind. Die Sprache verwendet ein klassisches Textformat zur Beschreibung der Regeln, welche, analog zu WS-RePolicy, auf dem ECA-Paradigma basieren.

4.5 ZUSAMMENFASSUNG

In diesem Kapitel wird die WS-Re2Policy-Sprache eingeführt sowie ihre Grundlagen und Anwendungsfelder erläutert. WS-Re2Policy stellt eine Spezifikationssprache dar, die es erlaubt, Anforderungen an Dienste sowie Reaktionen auf Abweichungen gleichzeitig zu modellieren. Bisherige Ansätze zur Spezifikation von Anforderungen sehen zumeist nur die Anforderungsdefinition vor. Die Behandlung von Abweichungen wird nicht weitergehend betrachtet, auch wenn deren Bedeutung erkannt und als Gegenstand möglicher Weiterentwicklungen genannt wird.

WS-Re2Policy orientiert sich am WS-Policy-Framework des W3C und kann auf dieser Grundlage Anforderungsbeschreibungen integrieren, die in einer beliebigen auf WS-Policy basierenden Sprache beschrieben sind. WS-Re2Policy fungiert dabei als Container für entsprechende Anforderungssprachen und erweitert diese um einfache Kontrollstrukturen zur Steuerung der Behandlung von Abweichungen. Da WS-

Re2Policy keine semantischen Annotationen unterstützt, ist es dem Endsystem überlassen, die Policy-Dokumente geeignet zu interpretieren.

Ein Anwendungsfeld für die Spezifikationssprache ist das im nächsten Kapitel vorgestellte AMAS.KOM Framework, welches WS-Re2Policy zur Definition des Handlungsspielraums einzelner teilautonom agierender Überwachungs- und Steuerungseinheiten verwendet.

5

EIN FRAMEWORK FÜR DIE VERTEILTE ÜBERWACHUNG UND STEUERUNG

In diesem Kapitel wird das *Automated Monitoring and Alignment of Services* Framework vorgestellt und Möglichkeiten zu dessen Umsetzung diskutiert. Das Framework adressiert unterschiedliche Aspekte eines verteilten Überwachungs- und Steuerungsansatzes. Zunächst werden ein Vorgehensmodell sowie ein zugehöriger Transformationsprozess definiert, mit deren Hilfe aus einem annotierten Workflow eine überwachte Instanz des Workflows generiert werden kann, für welchen die Überwachung und Steuerung verteilt erfolgen kann. Weiterhin definiert das Framework einen Architekturentwurf für die Realisierung eines auf dem Framework basierenden Systems unter Verwendung aktueller Technologien.

Das nachfolgende Kapitel ist wie folgt strukturiert: Zunächst werden im konzeptionellen Teil des Kapitels die Anforderungen an ein verteiltes Überwachungs- und Steuerungssystem im Kontext dienstbasierter Workflows definiert. Auf dieser Basis werden im nächsten Abschnitt zunächst die notwendigen Transformationsschritte beschrieben, die von einer annotierten Workflow-Beschreibung hin zu einer überwachten Instanz des Workflows führen, bevor danach eine die Anforderungen und Transformationsschritte unterstützende Architektur entwickelt wird. Im anschließenden Teil des Kapitels wird die Umsetzung des Frameworks auf Basis der Webservice- und Softwareagenten-Technologie vorgestellt und im Hinblick auf die durch die Lösung eingeführten Performanzveränderungen untersucht. Den Abschluss des Kapitels bilden die Diskussion der verwandten Arbeiten sowie eine Zusammenfassung des Kapitels.

5.1 ANFORDERUNGEN AN EIN FRAMEWORK ZUR VERTEILTEN ÜBERWACHUNG UND STEUERUNG

Wie bereits in vorangegangenen Abschnitten diskutiert, adressiert die Überwachung und Steuerung von dienstbasierten Systemen die Überprüfung und Gewährleistung der Qualität eines dienstbasierten Systems zu dessen Laufzeit. Auf Grund der möglichen Dynamik eines dienstbasierten Systems, in welchem Dienste zur Laufzeit ausgetauscht werden können, sind klassische Ansätze zur Überwachung und Steuerung häufig nicht anwendbar [Sero5]. Gerade die Aspekte der losen Kopplung von Diensten sowie die Berücksichtigung unterschiedlicher Kontrollsphären von Dienstanbietern, -nutzern sowie beteiligten Intermediären stellen besondere Anforderungen an die Überwachung und Steuerung eines dienstbasierten Systems.

Die Überwachung und Steuerung von Systemen besteht generell aus drei abstrakten Aktivitäten, welche nachfolgend aufgeführt sind (in Anlehnung an [BGP08], [Sero5]). Diese Aktivitäten gilt es in jedem Fall innerhalb eines Überwachungs- und Steuerungssystems geeignet zu berücksichtigen.

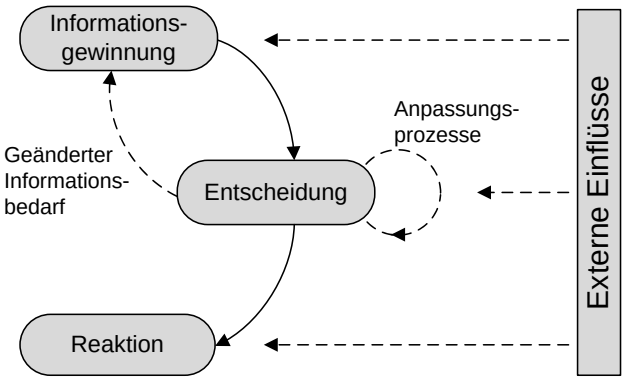


Abbildung 21: Grundlegende Überwachungs- und Steuerungsaktivitäten (in Anlehnung an [BGP08], [Ser05])

1. *Informationsgewinnung*: Gewinnung der für eine Entscheidungsfindung benötigten Informationen.
2. *Entscheidung*: Analyse der Informationen zur Feststellung von Abweichungen vom Sollzustand eines Systems.
3. *Reaktion*: Durchführen von Maßnahmen zur Behandlung von Fehlern und anderen Abweichungen.

Reine Überwachungsansätze bilden diese Aktivitäten ebenso ab wie kombinierte Ansätze zur gleichzeitigen Überwachung und Steuerung. Bei letzteren Ansätzen sind die Aktivitäten *Entscheidung* und *Reaktion* wesentlich komplexer auszuprägen.

Es gilt anzumerken, dass diese Aktivitäten unterschiedlichen Rückkopplungen unterliegen und externe Einflüsse auf sie einwirken. Abbildung 21 stellt auf einem hohen Abstraktionsgrad die Beziehung zwischen den einzelnen Aktivitäten sowie den vorhandenen Rückkopplungen dar. Rückkopplungen liegen unter anderem bei der Adaption an einen geänderten Informationsbedarf sowie bei der Anpassung der Entscheidungsfindung vor.

Aus den in Abbildung 21 dargestellten Rückkopplungen resultiert die Anforderung nach hoher Flexibilität des Überwachungs- und Steuerungsansatzes sowie dessen ausführenden Einheiten. Während ein geänderter Informationsbedarf im schlechtesten Falle ein geändertes Verfahren zu Informationsgewinnung impliziert, erfordern die Anpassungsprozesse auf Ebene der Entscheidungsfindung ein leicht anpassbares System zur Spezifikation von Anforderungen sowie deren Überprüfung. Ähnliche Anforderungen entstehen auch durch die Berücksichtigung von externen Einflüssen, d. h. den Einflussfaktoren, die nicht im Rahmen der Selbststeuerung des Systems entstehen.

Nachfolgend werden die Hauptanforderungen an den im Rahmen dieser Arbeit entwickelten Überwachungs- und Steuerungsansatz beschrieben, welcher die in Abbildung 21 dargestellten Überwachungs- und Steuerungsaktivitäten beinhaltet bzw. integriert. Die Anforderungen an ein entsprechendes Framework lassen sich hierbei in primär funktionale Anforderungen sowie nicht-funktionale Anforderungen unterscheiden:

- Primär funktionale Anforderungen an das Framework
  1. Unterstützung der Dynamik von dienstbasierten Systemen durch die Anpassbarkeit der Platzierung von Überwachungs- und Steuerungseinheiten sowie die Flexibilität der durch die Einheiten übernommenen Aufgaben.
  2. Berücksichtigung unterschiedlicher Kontrollsphären durch flexible Platzierung der Überwachungs- und Steuerungseinheiten.
  3. Unterstützung der gleichzeitigen Überwachung von Fehlern sowie Abweichungen von Anforderungen.
  4. Unterstützung aktueller Technologien zur Realisierung von Diensten, insbesondere der Webservice-Technologie, sowie Sicherstellung der Konformität zu den zugehörigen Standards.
  5. Erweiterbarkeit des Ansatzes durch eine flexible Architektur sowie Unterstützung eines geeigneten Erweiterungsmechanismus (Plugin-Konzept).
- Nicht-funktionale Anforderungen an das Framework
  6. Performanz des Ansatzes im Hinblick auf die Reaktionszeit auf Fehler und Abweichungen sowie Minimierung des durch den Ansatz entstehenden Overheads.
  7. Reduzierung des durch die Durchführung von Überwachungs- und Steuerungsmaßnahmen entstehenden Datenverkehrs zwischen Dienstnutzern, -anbietern sowie den Überwachungs- und Steuerungseinheiten.
  8. Gewährleistung von Skalierbarkeit und Robustheit des Ansatzes in Abgrenzung zu einem rein zentralen Ansatz.

Die Anforderungen lassen sich größtenteils direkt aus den Anforderungen und Beiträgen der vorangegangenen Kapitel ableiten. Die funktionalen Anforderungen 1 bis 3 lassen sich direkt aus der Motivation der Arbeit in Abschnitt 1.1 sowie den Anforderungen aus Abschnitt 3.1 ableiten. Die Umsetzung von Anforderung 3 ist gleichermaßen Forderung aktueller Standards für das Service-Management im IT-Betrieb (vgl. [GC07]). Anforderungen 4 und 5 sind reine Entwurfsanforderungen, die der Sicherstellung der Integrationsfähigkeit in bestehende Ansätze dienen. Die nicht-funktionalen Anforderungen 6 bis 8 lassen sich aus der Zielsetzung für die Verteilungsstrategien in Abschnitt 3.1 ableiten.

5.2 VORGEHENSMODELL UND TRANSFORMATIONSPROZESS

In diesem Abschnitt wird das AMAS.KOM zu Grunde liegende Vorgehensmodell beschrieben (Abschnitt 5.2.1), in welches ein Transformationsprozess (Abschnitt 5.2.2) integriert ist, der eine Workflow- sowie eine Anforderungsbeschreibung in eine überwachte Workflowinstanz überführt.

5.2.1 *AMAS.KOM Vorgehensmodell*

Das AMAS.KOM Vorgehensmodell, dessen Hauptbestandteil der in Abschnitt 5.2.2 beschriebene Transformationsprozess ist, besteht grundsätzlich aus fünf Aktivitäten bzw. Teilprozessen. Das vollständige Vorgehensmodell auf Basis der BPMN ist in Abbildung 22 dargestellt. Das Vorgehensmodell wird dabei nur in Teilen durch

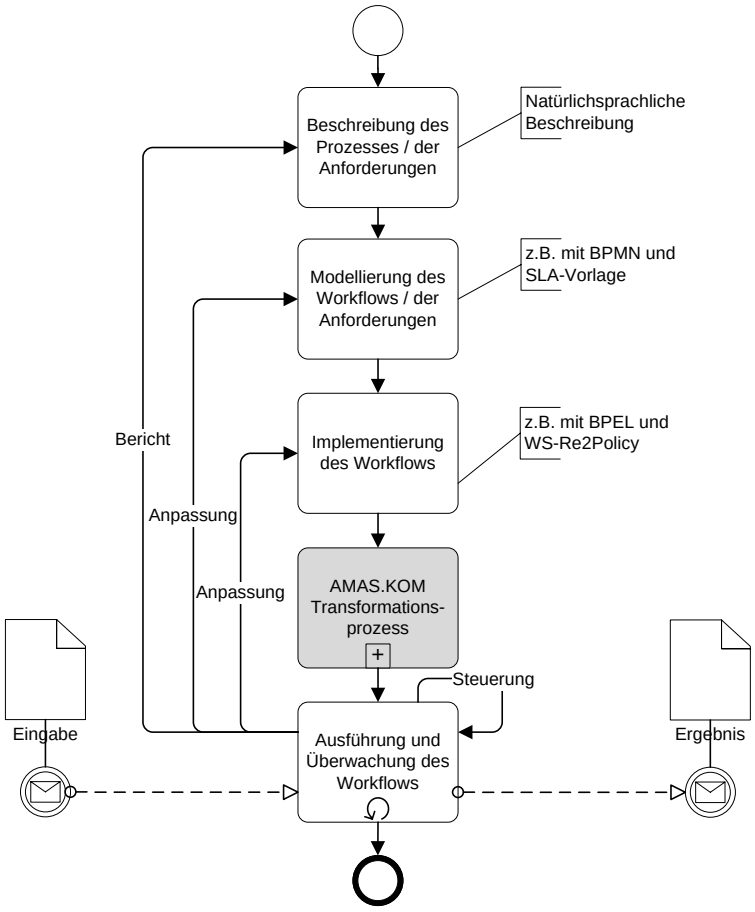


Abbildung 22: AMAS.KOM Vorgehensmodell

Werkzeuge oder automatisierte Transformationsverfahren unterstützt, da es gerade in den ersten Phasen des Vorgehensmodells einen hohen Interaktionsbedarf mit unterschiedlichen Nutzergruppen des zu überwachenden Workflows gibt, welche sich durch ihre Unstrukturiertheit auszeichnen.

Die erste der in Abbildung 22 aufgeführten Phasen beinhaltet die klassische Aufnahme und Dokumentation eines Geschäftsprozesses sowie der Anforderungen an diesen Prozess aus Sicht der Anwender innerhalb eines Unternehmens. Häufig wird zunächst der Ist-Zustand dokumentiert und vor Übergabe in die Modellierungsphase in den Soll-Zustand überführt. Innerhalb der Beschreibungsphase erfolgt die Dokumentation von Geschäftsprozessen sowie der zugehörigen Anforderungen in natürlicher Sprache, zumeist in Form eines einfachen Textdokuments. In der nachfolgenden Phase des Vorgehensmodells gilt es nun, die Beschreibungen in eine standardisierte Form zu überführen. Dabei kommen Modellierungsverfahren und -werkzeuge zum Einsatz, wie beispielsweise BPMN oder ereignisgesteuerte Prozessketten zur Modellierung der Prozesse. Anforderungen an die Prozesse werden in Form von SLA-Vorlagen bzw. Leistungsscheinen dokumentiert. Auf Basis der so gewonnenen Strukturierung kann innerhalb der nächsten Phase die Implementierung des Workflows, d. h. des durch IT unterstützbaren Teils eines Geschäftsprozesses, begonnen werden. Eine solche Implementierung erfolgt beispielsweise beim Einsatz der Webservice-Technologie unter Verwendung des WSBPEL-Standards, mit dessen

Hilfe der Workflow beschrieben und später ausgeführt werden kann. Bei der Realisierung des Workflows ist zu berücksichtigen, dass zur Implementierung der innerhalb des Workflows benötigten Funktionalitäten nur abstrakte Dienste (im Sinne von „Platzhaltern“ für konkrete Dienste – vgl. WSBPEL in Abschnitt A.1.2 des Anhangs) spezifiziert werden. Die Zuordnung konkreter Dienste zur abstrakten Workflowspezifikation erfolgt im Rahmen des Transformationsprozesses von AMAS.KOM. Zum Zwecke der Beschreibung der Anforderungen an den gesamten Workflow als auch an die einzelnen Dienste wird die in Abschnitt 4 vorgestellte WS-Re2Policy-Sprache verwendet, die beliebige WS-Policy-Sprachen zur Anforderungsdefinition verwenden kann. AMAS.KOM wertet hierbei allerdings nur solche Alternativen im Anforderungsteil des WS-Re2Policy-Dokuments aus, in welchen Dienstgüteanforderungen definiert sind. Im Transformationsprozess erfolgt auch die Überführung des angepassten Workflows in die Ausführungsumgebung, in der sie in der letzten Phase des Vorgehensmodells als überwachte Instanz des Workflows ausgeführt wird. Bei Verwendung der Webservice-Technologie erfolgt dies durch den Einsatz einer BPEL-Engine. Der Workflow verarbeitet im Betrieb bei fehlerfreiem Verhalten beliebig oft die vorgesehenen Eingaben und generiert das spezifizierte Ergebnis, ohne dass ein wiederholtes Durchlaufen des Vorgehensmodells notwendig wäre.

Das Vorgehensmodell sieht darüber hinaus eine Reihe von Rückkopplungen vor, die beim Auftreten von Fehlern oder Anforderungsabweichungen durchlaufen werden. Je nach Art und Schwere des aufgetretenen Fehlers bzw. der Abweichung werden unterschiedliche Eskalationsstufen mit den dazu gehörigen Maßnahmen durch das abstrakte Vorgehensmodell unterstützt. Solange mögliche Maßnahmen auf Basis der Entscheidungsbefugnis einer Überwachungs- und Steuerungseinheit oder auf Basis der Workflow-Engine getroffen werden können, liegt eine *Steuerung* der Dienste bzw. des Workflows vor. Fehler und Abweichungen können allerdings auch auf grundsätzliche Probleme in der Konzeption eines dienstbasierten Workflows zurückgeführt werden, so dass eine *Anpassung* auf anderer Ebene notwendig wird. Es besteht die Möglichkeit, dass die Implementierung des Workflows fehlerhaft ist oder auf Grundlage falscher Entwurfsentscheidungen durchgeführt wurde. Alternativ können auch die Modellierung des Prozesses oder die aus ihr abgeleiteten Anforderungen Anpassungsbedarf besitzen. Beide Anpassungsformen werden durch das Vorgehensmodell ausschließlich konzeptionell unterstützt. Darüber hinaus kann zu jedem Zeitpunkt der Ausführung ein ergänzender *Bericht* für den Anwender des Prozesses erstellt werden, auf dessen Basis grundlegende Anforderungen an einen Prozess oder sogar dessen Struktur angepasst werden können. Während die *Steuerung* automatisch durchgeführt wird, sind Anpassungen der Implementierung sowie der Modellierung zumeist nur manuell durchführbar. Sie finden als Bestandteil des Vorgehensmodells in AMAS.KOM Berücksichtigung, werden allerdings in der aktuellen Version des Frameworks nicht durch die bereitgestellte Architektur unterstützt.

5.2.2 *AMAS.KOM Transformationsprozess*

Hauptbestandteil des eben beschriebenen Vorgehensmodells ist der in diesem Abschnitt dargestellte Transformationsprozess. Der Transformationsprozess überführt die ihm übergebenen Workflow- und Anforderungsbeschreibungen in durch verteil-

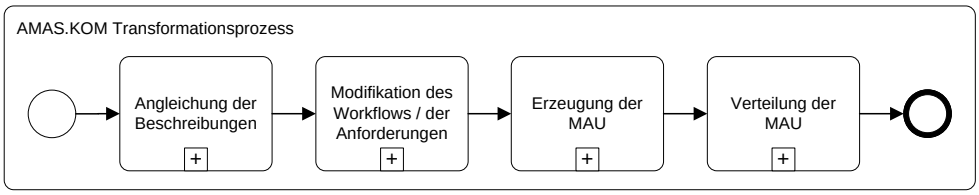


Abbildung 23: AMAS.KOM Transformationsprozess

bare MAU überwachte Workflow-Instanzen. Die Erläuterung des Transformationsprozesses erfolgt unter Zuhilfenahme der Webservice-Technologien WSBPEL und WS-Re2Policy.

Um eine überwachte Workflow-Instanz innerhalb des Frameworks zu erzeugen, sind eine Reihe von Schritten notwendig. Der abstrakte AMAS.KOM Transformationsprozess unterscheidet vier Schritte einer solchen Umwandlung, welche nachfolgend beschrieben werden. Abbildung 23 gibt einen Überblick über den abstrakten Transformationsprozess, wohingegen Abbildung 24 den Transformationsprozess am Beispiel des Einsatzes der Webservice-Technologie in Kombination mit der Softwareagenten-Technologie darstellt. Die einzelnen Schritte des Prozesses sind im Detail (vgl. [RES+08] und [Repo8]):[22]

1. *Angleichung der Beschreibungen* (Abbildung 24a)
2. *Modifikation des Workflows und der Anforderungen* (Abbildung 24b)
3. *Erzeugung der MAU* (Abbildung 24c)
4. *Verteilung der MAU* (Abbildung 24d)

Auf Grundlage einer beliebigen Beschreibung eines Workflows sowie der an den Workflow gestellten Anforderungen muss in einem ersten Schritt eine Überführung in eine für das weitere Verfahren verwendbare Form vorgenommen werden. Während der *Angleichung der Beschreibungen* werden die übergebenen Beschreibungsformen in WSBPEL und WS-Re2Policy überführt, sofern sie noch nicht in dieser Form vorliegen. Hierbei handelt es sich um einen manuellen Arbeitsschritt, der bei Vorliegen der korrekten Beschreibungsformen entfallen kann.

Unterstützt eine von WS-Re2Policy abweichende Beschreibungsform nicht die Spezifikation von Regeln zur Abweichungsbehandlung (vgl. Abschnitt 4), so werden an dieser Stelle Standardrichtlinien in die Beschreibung der Anforderungen übernommen. In diesem Schritt werden die Anforderungen an den Workflow durch eine einzige WS-Re2Policy-Datei vollständig beschrieben, welche durch die *RequirementsID* der WS-Re2Policy-Beschreibung einer Workflow-Beschreibung unter Verwendung einer externen Zuordnungstabelle zugeordnet werden kann. Die zu Grunde liegende WSBPEL-Beschreibung beinhaltet zu diesem Zeitpunkt die Zuordnung abstrakter Dienste.

22 Die Bezeichnung der Schritte in den zu Grunde liegenden englischsprachigen Veröffentlichungen [RES+08] und [Repo8] lauten *Annotation*, *Modification & Splitting*, *Generation* und *Distribution*.

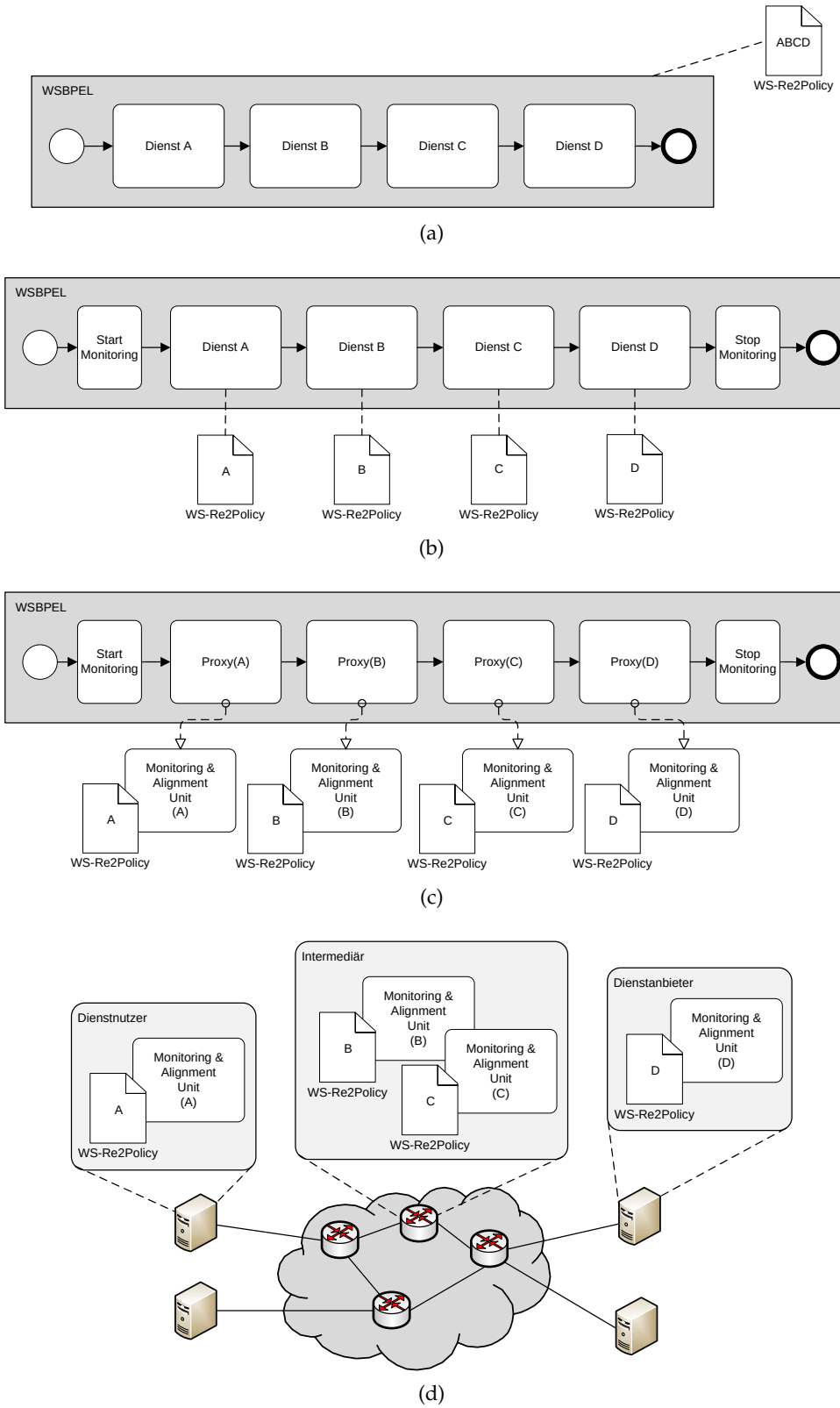


Abbildung 24: Beispiel einer Transformation

Im Schritt *Modifikation des Workflows und der Anforderungen* wird die Anforderungsbeschreibung für den gesamten Workflow analysiert und in Anforderungen an ein-

zelne Dienste zerlegt, die überwacht werden sollen. Basis hierfür sind die im vorangegangenen Schritt generierten Beschreibungen in WSBPEL und WS-Re2Policy. Zur Realisierung einer Zerlegung der Anforderungen sowie der damit verbundenen Zuordnung von konkreten Diensten zur Workflow-Beschreibung kommen Verfahren aus dem Bereich der dienstgüteorientierten Komposition zum Einsatz, wie sie z. B. von [BGR+07] und [CPEV05] beschrieben werden. Ihre Behandlung ist nicht Gegenstand dieser Arbeit. Dementsprechend finden Sie auch nicht in der AMAS.KOM Architektur Berücksichtigung. Ergebnisse der Anwendung der Verfahren sind Kompositionen von Diensten bzw. Ausführungspläne, welche die an den Workflow gestellten Anforderungen erfüllen. Nach Abschluss des Schrittes liegen somit ein Ausführungsplan mit einer Zuordnung konkreter Dienste in WSBPEL sowie eine Beschreibung der Anforderungen und möglichen Reaktionen auf Abweichungen in WS-Re2Policy auf Ebene von Einzeldiensten vor. Ebenfalls wird der ursprüngliche Workflow um zwei Dienstaufrufe erweitert, mit deren Hilfe bei der Workflow-Ausführung das Überwachungs- und Steuerungssystem gestartet und gestoppt werden kann. Zu diesem Zeitpunkt ist die Workflow-Beschreibung bereit zur Ausführung durch eine BPEL-Engine.

Die beiden nachfolgenden Schritte werden nicht nur bei einer erstmaligen Transformation durchgeführt, sondern können auch im laufenden Betrieb bei Anpassungsbedarf durchlaufen werden. Darüber hinaus finden sich Teile davon, wie z. B. die Ermittlung von Verteilungen, in den Überwachungs- und Steuerungseinheiten selbst wieder.

Während des Schritts *Erzeugung von MAU* werden auf Basis der Dienstauswahl sowie des zugehörigen Anforderungsdokuments Proxy-Komponenten erstellt bzw. bestehende Komponenten konfiguriert, welche als Stellvertreter die Dienstaufrufe annehmen und an das Überwachungs- und Steuerungssystem weiterleiten. Parallel dazu werden die benötigten MAU erzeugt und anhand der Anforderungen konfiguriert. Um das wiederholte Erzeugen von Konfigurationen zu vermeiden, ist es möglich, Konfigurationen von MAU für die jeweiligen Dienste wiederzuverwenden. Zu diesem Zweck wird innerhalb der im folgenden Abschnitt beschriebenen Architektur eine Datenhaltungskomponente integriert, welche vor Initiierung des Erzeugungsprozesses abgefragt wird.

Im selben Schritt erfolgt die Berechnung der Verteilung von MAU anhand der in Abschnitt 3 vorgestellten Verfahren. Zur Bestimmung der späteren Platzierung einer MAU durch Anwendung der Verteilungsstrategien wird zu Beginn auf von extern zur Verfügung gestellte Topologieinformationen zurückgegriffen. Bei einer wiederholten Durchführung der Verteilungsplanung durch die Einheiten selbst kommen eigens erhobene Topologieinformationen zum Einsatz. Die Ergebnisse der Verteilungsplanung können ebenfalls zum Zwecke der Wiederverwendung archiviert werden.

Abschließend erfolgt im Schritt *Verteilung von MAU* das eigentliche Deployment innerhalb der Infrastruktur, in dem auf Basis der Ergebnisse der vorangegangenen Lokationsplanung eine Verteilung der MAU durchgeführt werden. Eine Verteilung setzt eine geeignete Unterstützung der Infrastrukturkomponenten voraus. Am Beispiel der in Abschnitt 5.4 vorgestellten Implementierung auf Basis von Java sowie dem Agentenframework JADE (*Java Agent Development Framework* – siehe Abschnitt

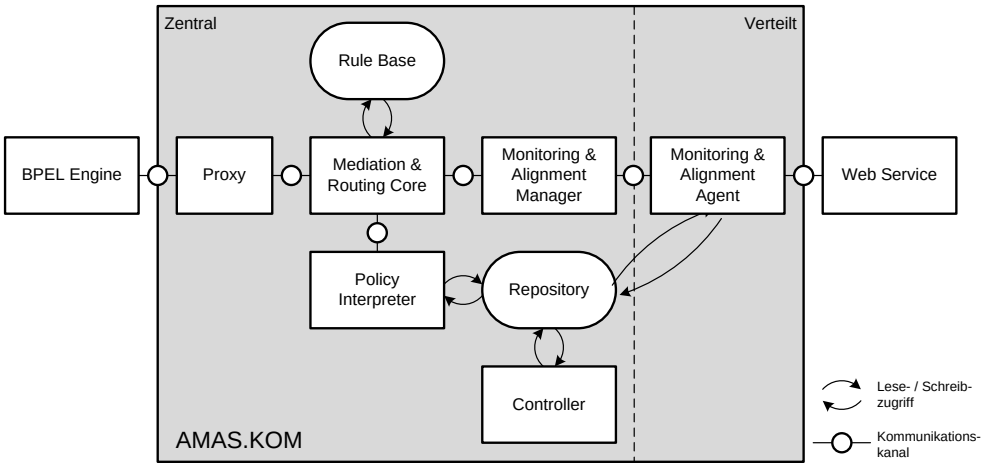


Abbildung 25: Architekturübersicht AMAS.KOM

A.2.2 des Anhangs) ist hierfür die Verfügbarkeit einer virtuellen Maschine für Java auf den Infrastrukturkomponenten Voraussetzung.

5.3 ARCHITEKTUR

In diesem Abschnitt wird der AMAS.KOM Architekturentwurf vorgestellt, welcher im Sinne einer Referenzarchitektur ein wichtiger Bestandteil des gesamten Frameworks ist. Der Abschnitt ist wie folgt gegliedert: Zunächst wird in Abschnitt 5.3.1 der Aufbau der Architektur auf Basis eines Komponentenmodells vorgestellt, bevor in Abschnitt 5.3.2 das Zusammenspiel der Komponenten während der Ausführung einer Implementierung der Architektur im Detail betrachtet wird.

Für die Gestaltung der Architektur gelten dieselben Anforderungen, die bereits zuvor in Abschnitt 5.1 für das gesamte Framework definiert wurden.

5.3.1 *Aufbau der AMAS.KOM Architektur*

Die AMAS.KOM zu Grunde liegende Architektur besteht aus einer Reihe von Komponenten, welche in diesem Abschnitt vorgestellt werden. Eine Architekturübersicht auf FMC-Basis ist Gegenstand von Abbildung 25. Die Architektur ist eine Fortentwicklung der in [RES+08] und [Repo8] vorgestellten Architekturentwürfe. Der vorliegende Architekturentwurf gruppiert verschiedene Komponenten aus Gründen einer effizienteren Implementierung.

Die in Abbildung 25 repräsentierten Komponenten lassen sich in Kernkomponenten, unterstützende Komponenten sowie Fremdkomponenten unterscheiden. Die Kernkomponenten bilden die Funktionalitäten ab, welche für die Arbeitsweise des AMAS.KOM Frameworks kennzeichnend sind. Diese Kernkomponenten sind:

- *Mediation & Routing Core*: Die Komponente ist für die Steuerung der Kommunikation zwischen den Komponenten der Architektur zuständig. Durch sie werden Dienstaufrufe, die vom *Proxy* stammen, auf verfügbare *Monitoring Manager*

geleitet. Darüber hinaus selektiert sie die notwendigen Konfigurationsinformationen und Politiken, die einem Dienstaufruf zugeordnet werden können.

- *Monitoring Manager*: Der *Monitoring Manager* ist für die Konfiguration der Überwachungs- und Steuerungseinheiten verantwortlich und übernimmt ebenfalls die Planung der Platzierung der Einheiten innerhalb der Infrastruktur.

- *Policy Interpreter*: Die Komponente verarbeitet die Politiken, die zu den verschiedenen Diensten innerhalb des *Repository* zuvor hinterlegt wurden. Sollten für einen Dienst verschiedene sich widersprechende Politiken vorhanden sein, so berechnet die Komponente die effektive Politik, d. h. die Politik, die im aktuellen Fall angewendet werden soll.[23]

- *Monitoring & Alignment Agent*: Die Komponente bildet das Konzept der in der bisherigen Arbeit als MAU bezeichneten Überwachungs- und Steuerungseinheit ab. Sie ist für den eigentlichen Dienstaufruf sowie für die Überwachung und Steuerung des Dienstes im Rahmen des ihr durch die gültige Politik eingeräumten Handlungsspielraums verantwortlich.

- *Controller*: Der *Controller* stellt die Schnittstelle zu externen Workflow- und Anforderungsdefinitionen dar. Mit Hilfe dieser Schnittstelle können die relevanten Daten an das AMAS.KOM System übergeben werden. Innerhalb der Komponente erfolgen darüber hinaus die eigentliche Transformation der Workflow-Beschreibung sowie die Aufteilung der Anforderungsdokumente auf Einzeldienste.

Wie die Bezeichnung *Monitoring & Alignment Agent* (MAA) bereits andeutet, erfolgt die Umsetzung der *Monitoring & Alignment Units* auf Basis der in Abschnitt A.2 beschriebenen Agententechnologie. Die Auswahl der Agententechnologie erfolgt hierbei auf Grund der Existenz geeigneter technischer Frameworks in diesem Anwendungsfeld zur Kapselung und Verteilung der benötigten Überwachungs- und Steuerungsfunktionen. Die Auswahl ist weniger durch das Vorliegen eines unstrukturierten und evtl. nicht-deterministischen Lösungsraums in komplexen Problemfeldern bedingt, welches Lockemann und Nimis in [LN04] als möglichen Indikator für den Agenteneinsatz identifizieren. Agenten dienen vielmehr als technologische Plattform zu Implementierungszwecken, d. h. es werden nur einzelne der durch die Agententechnologie ermöglichten Eigenschaften genutzt (siehe Abschnitt A.2.1 des Anhangs). MAA sind reaktiv, da sie nach Auswertung von Messergebnissen auf sich ändernde Situationen mit Gegenmaßnahmen reagieren. Sie handeln so lange autonom, wie es der vordefinierte Handlungsspielraum erlaubt und sind im Normalfall persistent, d. h. sie existieren auch zwischen zwei Dienstaufrufen weiter. Die Kollaboration von MAA zur gemeinsamen Problemlösung wird in Abschnitt 6.2 als Erweiterungsmöglichkeit beschrieben, wohingegen die Proaktivität von Agenten keinen Einsatz findet.

Als Nächstes soll an dieser Stelle auf den Aufbau des *Monitoring & Alignment Agent* eingegangen werden, welcher nicht in allen Aspekten dem klassischen Aufbau eines

23 Im Standardfall wird die Politik ausgewählt, welche die stärksten Restriktionen aufweist. Weiterhin besteht die Möglichkeit, Schnittmengen der einzelnen Politiken zu bilden, sofern diese miteinander vereinbar sind. Die Bestimmung der effektiven Politik ist allerdings nicht Gegenstand dieser Arbeit.

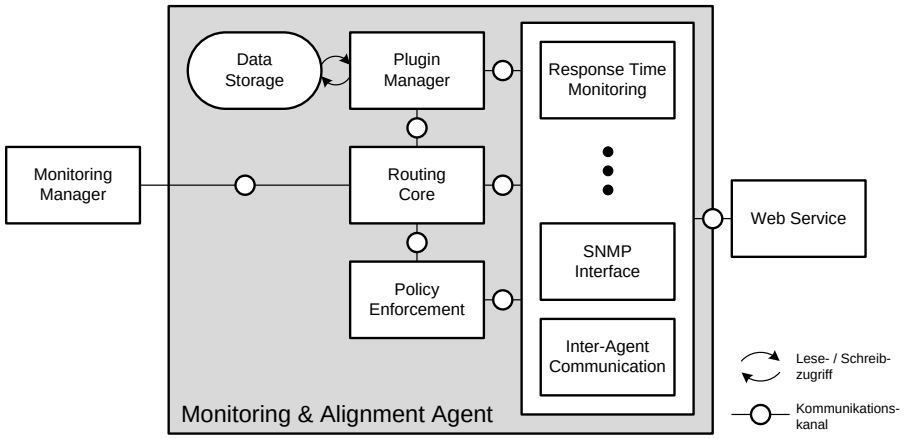


Abbildung 26: Aufbau eines Monitoring & Alignment Agents

Softwareagenten entspricht, sondern um eine Reihe zusätzlicher Komponenten erweitert wurde. Um einen hohen Grad an Flexibilität in Bezug auf zu überwachende Eigenschaften sowie unterstützte Steuerungsmaßnahmen sicherzustellen, sind MAA intern ebenfalls modular im Sinne eines dienstbasierten Systems aufgebaut. Auf Basis einer Form der losen Kopplung lassen sich einzelne Dienste zur Messung zusätzlicher Dienstgütemerkmale, zur Erweiterung der Entscheidungsmechanismen eines Agenten sowie Dienste für die Kommunikation mit der Außenwelt zur Laufzeit ergänzen. Abbildung 26 stellt den Aufbau eines solchen Agenten dar. Zu dessen Kernkomponenten zählen der *Plugin Manager*, der für die Verwaltung der Erweiterungsdienste zuständig ist und diese innerhalb des MAA zur Verfügung stellt, der *Routing Core*, der den Dienstaufruf den relevanten internen Komponenten zuordnet, sowie das *Policy Enforcement*, welches über die Einhaltung der Politiken wacht und bei Bedarf Korrekturmaßnahmen initiiert.

Im Detail sind im aktuellen Architekturentwurf für den MAA drei Kategorien von Erweiterungsdiensten vorgesehen. Dienste aus der Kategorie *Kontrolllogik* beschäftigen sich mit den Aspekten der Datenanalyse und der Entscheidung auf Basis der analysierten Daten sowie mit Entscheidungen über die Relokation des MAA innerhalb der Infrastruktur. Die Gewinnung von Messdaten unterschiedlichster Art sowie die Einbindung bestehender Überwachungsansätze, wie z. B. SNMP, sind Aufgaben der Dienste aus der Kategorie *Informationsgewinnung*. Von besonderer Relevanz ist hierbei die Gewinnung von Topologieinformationen zur Bestimmung von Verteilungen für MAA innerhalb einer Infrastruktur. In die Kategorie *Kommunikation* fallen die Dienste, die sich mit dem aktiven Austausch von Daten mit externen Systemen beschäftigen. Hierzu zählt die Integration der MAA in Systeme auf Basis des Web Service Distributed Management Standards (WSDM – vgl. Abschnitt 5.6.3) ebenso wie die Interaktion zwischen MAA.

In Ergänzung zu den Kernkomponenten werden weitere unterstützende Komponenten benötigt, welche für eine Implementierung des AMAS.KOM Frameworks wichtige Funktionen anbieten. Allerdings handelt es sich hierbei zumeist um Standardfunktionalitäten, die innerhalb des AMAS.KOM Architekturentwurfs nicht erneut spezifiziert bzw. entwickelt werden. Zu diesen Komponenten zählen in Abbildung 25 die folgenden Komponenten:

- *Proxy*: Der *Proxy* wird durch den modifizierten Workflow aufgerufen und leitet den Dienstaufruf auf die Überwachungs- und Steuerungsinfrastruktur um.
- *Rule Base*: Die Regeldatenbank wird verwendet, um Konfigurationsinformationen für die *Mediation & Routing Core* zu verwalten. In ihr werden Zuordnungen zwischen Diensten und für die Dienst zuständigen *Monitoring Manager* ebenso abgelegt, wie die Zuordnungsregeln von Politiken zu Dienstaufrufen.
- *Repository*: Das *Repository* repräsentiert die zweite Datenhaltungskomponente innerhalb der Architektur. In der Komponente werden Konfigurationen von *Monitoring Manager* sowie *Monitoring & Alignment Agent* inkl. deren Lebenszyklusinformationen verwaltet. Darüber hinaus werden im *Repository* sämtliche Politiken verwaltet und zum Zwecke der Wiederverwendung zur Verfügung gestellt.

Fremdkomponenten sind kein integraler Bestandteil der in Abbildung 25 dargestellten Architektur, werden aber von einer zu der AMAS.KOM Architektur konformen Implementierung genutzt bzw. nutzen diese Implementierung selbst. Zu den Fremdkomponenten zählen die konsumierten Dienste sowie die BPEL-Engine, welche für die Ausführung des Workflows eingesetzt wird.

Die Verlässlichkeit der MAA als Kernbestandteil der AMAS.KOM Architektur ist selbst nicht Gegenstand dieser Arbeit, stellt aber gerade in kooperativen Szenarien eine Herausforderung dar, in welchen Agenten gemeinsam an Überwachungs- und Steuerungsaufgaben arbeiten (siehe hierzu Abschnitt 6.2). Eine Behandlung des Themas Verlässlichkeit von Softwareagenten aus architektonischer Sicht wird in [LN06] auf Basis einer Referenzarchitektur für Softwareagenten (vgl. [LN05]) beschrieben. Die Integration der vorgestellten Konzepte ist eine zukünftige Erweiterungsmöglichkeit der AMAS.KOM Architektur.

5.3.2 *Interaktion der Komponenten zur Laufzeit*

Auf Basis der im vorangegangenen Abschnitt vorgestellten Komponenten wird in diesem Abschnitt die Interaktion der Komponenten bei der Ausführung eines auf der Architektur basierenden Systems diskutiert. Für eine Detaildarstellung der Interaktionen in Form eines UML-Sequenzdiagramms wird auf Abbildung 39 in Abschnitt A.6.2 des Anhangs zu dieser Arbeit verwiesen.

Grundsätzlich beginnt die Interaktion mit einem auf AMAS.KOM basierenden System durch einen vom *Proxy* entgegengenommenen Dienstaufruf. Der Dienstaufruf selbst stammt in diesem Fall von einer BPEL-Engine, welche als Dienstnutzer einen Workflow abarbeitet. Der *Proxy* fungiert dabei als Stellvertreter und Schnittstelle nach außen für die restlichen Komponenten. Der *Proxy* ist selbst nach außen hin über eine Dienstschnittstelle ansprechbar und wird als Stellvertreter für die eigentlich aufzurufenden Dienste in den zu überwachenden Workflow integriert. Bei Verwendung der Webservice-Technologie und WSBPEL zur Beschreibung des Workflows wird die Workflow-Beschreibung im Rahmen der Transformation so verändert, dass neben der Integration zweier Dienstaufrufe zum Starten und Stoppen des Überwachungs- und Steuerungssystems die Webservice-Aufrufe durch parametrisierte Aufrufe des Proxy ersetzt werden. Der *Proxy* leitet den Aufruf an den *Media-*

tion & Routing Core weiter, der diesen analysiert und an die nachgelagerten Komponenten weiterleitet. Nachdem der Dienstaufruf erfolgreich durch die nachgeschalteten Komponenten durchgeführt wurde, werden die Ergebnisse des Dienstaufrufs anschließend über den *Proxy* an die aufrufende Instanz zurückgegeben.

Der *Mediation & Routing Core* entscheidet im nächsten Schritt, welcher reale Dienst im vorliegenden Fall aufzurufen ist. Der Dienstaufruf, welcher durch die BPEL-Engine als Dienstnutzer vorgenommen wurde, besteht zunächst nur aus dem Aufruf der Proxy-Komponente bei gleichzeitiger Übermittlung der Identifikation des eigentlich aufzurufenden Dienstes. Der *Mediation & Routing Core* rekonstruiert somit den ursprünglichen Dienstaufruf und ordnet diesem zusätzlich die hinterlegten Anforderungsspezifikationen bzw. bereits vorhandenen MAA-Konfigurationen zu. Hierzu werden die relevanten Politiken aus dem *Repository* ausgewählt und ausgewertet. Notwendige Konfigurationsdaten werden durch die *Rule Base* zur Verfügung gestellt. Im Falle mehrerer unterschiedlicher vorliegender Politiken für einen Dienstaufruf ermittelt der *Mediation & Routing Core* unter Einbezug des *Policy Interpreter* die effektive Politik. Danach übergibt der *Mediation & Routing Core* die Kontrolle des Dienstaufrufs zusammen mit den in den nachfolgenden Schritten benötigten Informationen an den verantwortlichen *Monitoring Manager*.

Der *Monitoring Manager* erzeugt auf Basis der ihm übergebenen Informationen die *Monitoring & Alignment Agents* oder wählt bestehende MAA aus, welche den aktuellen Dienstaufruf übernehmen können. Ein solcher MAA kann individuell bei jedem Dienstaufruf erstellt werden, wobei wirklicher Nutzen erst aus der Wiederverwendung von MAA bzw. deren Konfigurationen und Platzierungen resultiert. Der *Monitoring Manager* sorgt ebenfalls für die erstmalige Verteilung der MAA innerhalb der Infrastruktur. Der MAA ruft den Dienst auf und überwacht dessen Aufruf. Die Ergebnisse der Überwachung können im *Repository* für spätere Auswertungszwecke abgelegt werden. Beim Aufruf wird versucht, die dem Dienst zugeordnete Politik zu erfüllen, d. h. den Dienstaufruf im Einklang mit den vorgegebenen Rahmenbedingungen und unter Berücksichtigung der Handlungsspielräume erfolgreich durchzuführen. Nachdem die Politik erfüllt wurde, wird die Kontrolle wieder zurück an den *Proxy* gegeben, der die aufrufende Instanz kontaktiert. Es wird in jedem Fall ein Ergebnis zurückgeliefert, auch wenn Fehler oder Abweichungen von den Anforderungen festgestellt wurden, die nicht durch den MAA direkt behoben werden konnten. Eine Politik ist selbst dann erfüllt, wenn ein Dienstaufruf nicht erfolgreich war, sofern die durch die Politik definierten Korrekturmaßnahmen durchgeführt wurden und somit der Handlungsspielraum des MAA erschöpft ist.

5.4 TECHNISCHE UMSETZUNG UND IMPLEMENTIERUNG

Während in den vorangegangenen Abschnitten das AMAS.KOM aus konzeptioneller Sicht vorgestellt wurde, so wird in diesem Abschnitt eine konkrete Implementierung für das AMAS.KOM Framework auf Basis der Webservice-Technologie sowie mobiler Softwareagenten präsentiert. Die Darstellung in diesem Abschnitt konzentriert sich dabei auf sämtliche Elemente des Frameworks, welche direkt in die Überwachung und Steuerung eines Dienstes involviert sind. Aus Komplexitätsgründen wird auf die notwendigen Interaktionen mit einem Workflow-Nutzer, die Modellie-

rung von Workflows und Politiken sowie das Management von Politiken in diesem Abschnitt nicht weiter eingegangen.

Bevor einzelne Umsetzungsdetails vorgestellt werden, soll zunächst ein Überblick über die zum Einsatz kommenden Technologien und Standards gegeben werden. Die Umsetzung basiert auf etablierten Webservice-Standards (siehe Abschnitt A.1.2), wie dem Einsatz von SOAP in Version 1.2 als Transportprotokoll, WSDL in Version 1.1 zur Beschreibung der Schnittstellen der Dienste sowie WSBPEL 2.0 zur Beschreibung der dienstbasierten Workflows. Als unterliegender Transportmechanismus für SOAP kommt HTTP zum Einsatz. Für die Umsetzung der MAA wird das Softwareagentenframework JADE in Version 3.5 eingesetzt, welches eine Referenzimplementierung des zuvor vorgestellten FIPA-Standards (*Foundation for Intelligent, Physical Agents* – vgl. Abschnitt A.2.2) darstellt. Zur Beschreibung von Anforderungen und Reaktionen auf Anforderungsabweichungen kommt die WS-Re2Policy-Sprache zum Einsatz (siehe Kapitel 4).

Der erste Ausführungsschritt in einer auf AMAS.KOM basierenden Implementierung ist die Transformation eines BPEL-basierten Workflows in eine überwachte Instanz desselben Workflows. Anders als bei den Ansätzen aus verwandten Arbeiten (vgl. z. B. Baresi et al. in Abschnitt 5.6) wird eine Workflow-Beschreibung nicht unter Einbeziehung der zugehörigen Anforderungsdefinitionen in eine überwachte Instanz transformiert, sondern vollständig unabhängig von dieser Definition erstellt. Die Zuordnung einer oder mehrerer Anforderungsdefinitionen erfolgt erst zur Laufzeit, was den Austausch von Anforderungsdefinitionen ohne erneute Transformation einer Workflow-Beschreibung ermöglicht. Während der Transformation des BPEL-Dokuments wird jeder Dienstaufruf durch den Aufruf des *Proxy* ersetzt, welcher alle für den Aufruf des eigentlichen Webservice notwendigen Parameter übergeben bekommt. Hierbei werden die *messageTypes* sowie der *portType* der BPEL-Beschreibung ersetzt sowie eine Reihe von *assign* Befehlen integriert, die die Übergabeparameter des ursprüngliche Dienstes an den *Proxy* übergeben sowie nach Beendigung des Dienstes diese wieder in die der ursprünglichen BPEL-Beschreibung entsprechende Form zurück transformieren. Entsprechende Anpassungen innerhalb der zugehörigen WSDL-Beschreibung werden analog vorgenommen. Zur Verarbeitung der BPEL- und WSDL-Beschreibungen kommen eigene Programmroutinen in Kombination mit der XML Path Language (XPath) zum Einsatz. XPath verwendet hierbei das Document Object Model (DOM).[24]

Bei der Ausführung eines transformierten Workflows erfolgt im nächsten Schritt der Aufruf des *Proxy* als Webservice, bei dem alle notwendigen Parameter zum Aufruf des eigentlich angefragten Webservices mit übergeben werden. Diese Informationen werden nachfolgend an den *Mediation & Routing Core* weitergeleitet, wobei gleichzeitig auch die Übergabe einer Host-Adresse des *Proxy* sowie ein dem Dienstaufruf zugeordneter Port mit übergeben wird. Diese werden benötigt, um die Rückgabewerte vom MAA zum *Proxy* zu übermitteln. Der Proxy öffnet hierfür einen Socket mit der gewählten Portnummer. Dies ist notwendig, da sonst keine Zuordnung zwischen dem Agenten und dem *Proxy* möglich ist, da MAA nur indirekt durch den *Proxy* über die *Monitoring Manager* aufgerufen werden. Zur Vergabe der Portnummern wird hierbei ein zufälliges Vergabeverfahren gewählt [OW02]. Jeder

24 Für weitere Informationen zu XPath sowie DOM wird an dieser Stelle auf [CD99] und [KPRR03] verwiesen.

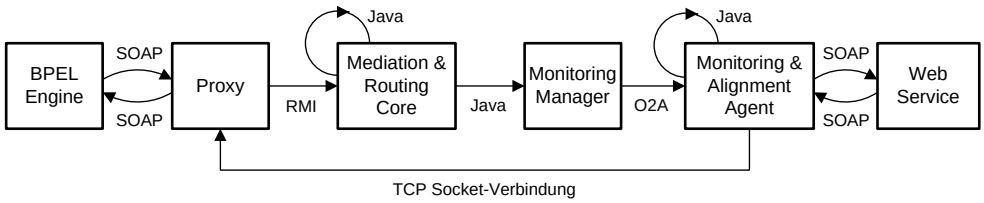


Abbildung 27: Technologien bei der Interaktion der Komponenten

der Dienstaufrufe wird auf Seiten des *Proxy* innerhalb eines eigenen Threads verarbeitet.

Der *Proxy* kontaktiert nachfolgend unter Verwendung der Remote Method Invocation (RMI) von Java den *Mediation & Routing Core*. Eine Übersicht aller Technologien, die bei der Interaktion der AMAS.KOM Komponenten involviert sind, wird in Abbildung 27 dargestellt. Der *Mediation & Routing Core* fragt die effektive Politik für den aufgerufenen Dienst beim *Policy Interpreter* ab sowie, sofern notwendig, weitere Konfigurationsinformationen bei der *Rule Base*. Aus Performanzgründen erfolgt der Aufruf wiederum durch die Nutzung Java-interner Aufrufmechanismen und nicht durch den Einsatz von Webservice-Technologien. Die so zusammengestellten Informationen werden an den *Monitoring Manager* weitergereicht.

Bei der Übergabe des Dienstaufrufs sowie der zusätzlichen Informationen an den *Monitoring Manager* wird die JADE-Agentenplattform initialisiert. Der *Monitoring Manager* ist für die Verwaltung der benötigten MAA zuständig. Da er vollständig auf der JADE-Plattform basiert, werden die durch JADE eingeführten Konzepte zum Management von Agenten eingesetzt (vgl. [BCG07]). Innerhalb eines JADE-basierten Systems können Agenten niemals außerhalb eines Containers existieren, der die Laufzeit- und Verwaltungsumgebung für die Agenten darstellt. Der *Monitoring Manager* stellt hierbei den so genannten *Main Container* dar, welcher die Verwaltung anderer Container übernimmt. Diese können innerhalb einer Infrastruktur auf Basis geeigneter Lokationsverfahren verteilt werden, müssen sich aber beim *Main Container* registrieren. Darüber hinaus initiiert der *Monitoring Manager* die MAA in den Containern und ist ebenfalls für die Verteilung der Container in der Infrastruktur zuständig. Im Rahmen der Initiierungsphase werden dem MAA dabei die vom *Mediation & Routing Core* empfangenen Informationen übergeben. Jeder Agent ist nach außen über ein *AgentController* Objekt ansprechbar, wobei eine direkte Interaktion zur Wahrung dessen Autonomie nicht möglich ist. Zur Kommunikation mit der Außenwelt wird hierbei das so genannte Object-to-Agent (O2A) Verfahren eingesetzt, wobei eine synchronisierte FIFO-Queue zum Austausch von Java-Objekten zur Kommunikation mit dem Agenten verwendet wird (siehe hierzu [BCG07]).

Der MAA kann einen Dienstaufruf aus der O2A-Queue erst dann entgegennehmen, wenn seine Initialisierung abgeschlossen ist. Zuvor können dem Agenten noch keine dienstspezifischen Informationen mitgegeben werden. Der *Monitoring Manager* und der MAA müssen aus diesem Grund synchronisiert werden. Der *Monitoring Manager* geht in einen Wartezustand über, nachdem der MAA initiiert wurde. Sobald dieser zurückmeldet, dass er fertig initiiert ist, kann der Dienstaufruf sowie die weiteren Konfigurationsinformationen in die Queue zur Abholung durch den MAA eingestellt werden. JADE-Agenten führen unter Verwendung des SOAP with Attachments for Java (SAAJ) Standards danach den eigentlichen Dienstaufruf durch. SAAJ

erlaubt es ohne vorherige Generierung von Stubs aus WSDL-Informationen die für die Interaktion mit einem Webservice notwendigen XML-Dokumente zu erstellen.

Die einzelnen Funktionen für den Dienstaufruf, der Plugins für die Messung und die Durchführung von Korrekturmaßnahmen sowie der möglichen Interaktion mit Fremdsystemen sind als *Behaviour* (Verhalten) im Sinne des JADE-Frameworks in den MAA implementiert.[25] Nach Abarbeitung des Dienstaufrufs und der möglichen Durchführung von Korrekturmaßnahmen erfolgt die Rückgabe des Ergebnisses direkt an den *Proxy* durch Verwendung der bekannten Kombination aus Host-Adresse und Portnummer.

Die Implementierung setzt die Kernfunktionalitäten des AMAS.KOM Frameworks um. Neben der Spezifikation der Anforderungen und Abweichungen auf Anforderungen sowie deren Verwaltung wird auch auf die Berechnung der Lokation des Agentencontainers bzw. des darin enthaltenen MAA nicht weiter eingegangen. Für die Diskussion dieser Inhalte wird auf die entsprechenden, vorangegangenen Kapitel verwiesen. Im Rahmen der Umsetzung wurden ebenfalls nicht alle möglichen Reaktionstypen implementiert, da beispielsweise die Komplexität von Replanning- und Renegotiation-Ansätzen eigenständige Forschungsbereiche darstellen.

5.5 EVALUATION DER PROTOTYPISCHEN UMSETZUNG

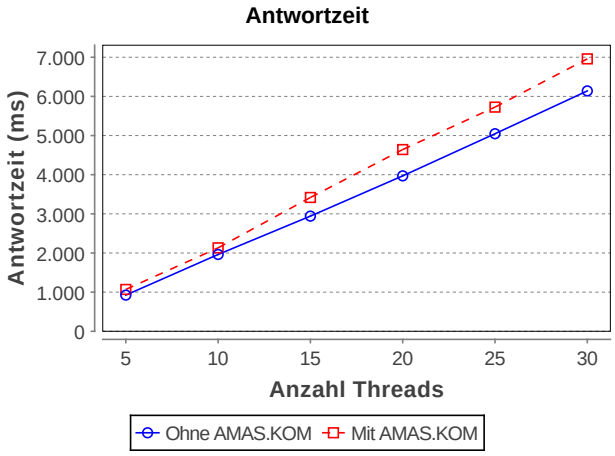
In diesem Abschnitt wird untersucht, wie sich die Integration eines auf AMAS.KOM basierenden Überwachungs- und Steuerungssystems auf die Performanz des zu Grunde liegenden Gesamtsystems auswirkt. Es wird analysiert, ob die durch Einführung des Systems entstehenden Performanzeinbußen den Nutzen einer Verteilung von MAU (siehe Abschnitt 3.6.3) beeinträchtigen. Hierzu werden der Durchsatz sowie die Antwortzeit des Dienstaufrufs anhand der im vorangegangenen Abschnitt vorgestellten Lösung diskutiert. Für weitere Information zur Untersuchung wird auf Abschnitt A.6.1 des Anhangs verwiesen.

Unter Verwendung der Implementierung aus Abschnitt 5.4 wird innerhalb eines Testbeds der Einfluss der Überwachung und Steuerung untersucht, wobei das Testbed neben einer AMAS.KOM Installation einen Standard-Applikationsserver als Dienstanbieter sowie das Apache JMeter Messwerkzeug als Dienstnutzer beinhaltet.[26]

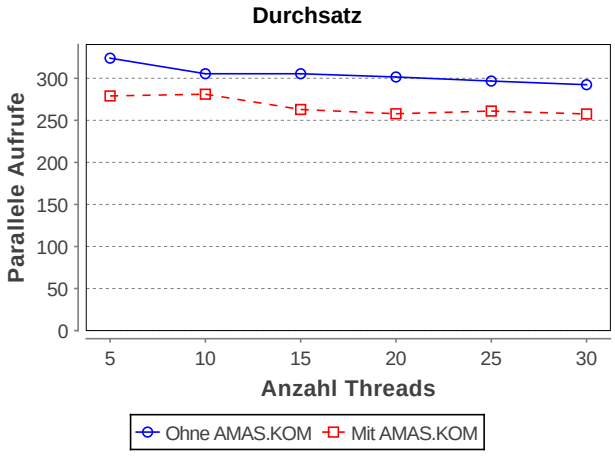
Im Detail wird innerhalb des Messwerkzeugs JMeter ein Testplan definiert, der unter Verwendung von HTTP direkt auf den Webservice des Dienstanbieters zugreift. Der bereitgestellte Webservice führt hierbei eine einfache Berechnung auf Basis des Aufrufparameters durch. Die Funktion des Webservice wird dabei bewusst einfach gehalten, um den Einfluss der Bearbeitungsdauer auf Seiten des Applikationsservers auf die Antwortzeit des Webservice vernachlässigbar gering zu halten. Der Testplan beinhaltet 6 unterschiedliche Testsets, die sich anhand der gleichzeitig durchgeführten Dienstaufrufe unterscheiden. Es werden im Rahmen der Untersuchung Tests mit 5, 10, 15, 20, 25 und 30 gleichzeitigen Threads, d. h. parallelen Dienstanfragen,

25 Die Implementierung entsprechender *Behaviours* erfolgt durch klassischen Java-Programmcode.

26 JMeter kommt in Version 2.3.2 zum Einsatz – weitere Informationen zu JMeter finden sich unter http://jakarta.apache.org/jmeter/ – abgerufen am 14.04.2009.



(a)



(b)

Abbildung 28: Ergebnisse der Performanzanalysen

durchgeführt. Die Anzahl der Dienstanfragen ist dabei durch eine vorgegebene Testhöchstdauer definiert. JMeter versucht, innerhalb des vordefinierten Zeitraums eine maximale Anzahl von Dienstaufrufen durchzuführen, über welche bei der Auswertung der Werte gemittelt werden kann. Eine detaillierte Darstellung der jeweiligen Stichprobengrößen erfolgt in Tabelle 23 in Abschnitt A.6.1 des Anhangs.

Abbildung 28a stellt die Ergebnisse der Untersuchung der Antwortzeit des Webservice dar, wohingegen in Abbildung 28b die Ergebnisse für den Durchsatz gleichzeitiger Anfragen in Relation zu den verschiedenen Thread-Konfigurationen abgebildet wird. Die zugehörigen Ergebnisse sind in den Tabellen 14 und 15 dokumentiert.

Die Messergebnisse zeigen für das Antwortverhalten des Dienstes ein eindeutig lineares Wachstum bei zunehmender Anzahl gleichzeitiger Zugriffe (siehe Abbildung 28a). Diese Aussage gilt sowohl für den Dienstaufruf ohne Überwachung als auch für den Aufruf des Dienstes über die AMAS.KOM Infrastruktur. Bei Analyse der durchschnittlichen Antwortzeiten lässt sich im Schnitt eine Verschlechterung der Antwortzeiten von rund 14 % (Tabelle 14) beobachten, die durch die Integration des Überwachungs- und Steuerungsansatzes begründet werden kann. Das lineare Wachstum der Antwortzeiten von sowohl unüberwachten als auch überwachten

Threads	5	10	15	20	25	30
Ohne AMAS	924	1.962	2.942	3.970	5.046	6.140
Mit AMAS	1.069	2.131	3.418	4.642	5.726	6.959
Veränderung (%)	15,69	8,61	16,18	16,93	13,48	13,34

Tabelle 14: Durchschnittliche Antwortzeiten (Angaben in ms)

Threads	5	10	15	20	25	30
Ohne AMAS	323,91	305,39	305,38	301,53	296,70	292,34
Mit AMAS	278,99	281,06	262,93	257,82	261,04	257,55
Veränderung (%)	-13,87	-7,97	-13,90	-14,50	-12,02	-11,90

Tabelle 15: Ergebnisse der Messung des Durchsatzes

Dienstaufrufen gibt ebenfalls einen Hinweis auf eine grundsätzliche Skalierbarkeit einer solchen Lösung.

Beim Vergleich der Messergebnisse des Durchsatzes der unüberwachten und überwachten Lösung lässt sich auch hier ein annähernd konstanter Satz der Verschlechterung des Durchsatzes durch Einführung von AMAS.KOM beobachten (vgl. Abbildung 28b und Tabelle 15). Die Verschlechterung des Durchsatzes durch die Integration von AMAS.KOM liegt hier im Durchschnitt bei 12 %. Die Konstanz weist auch hier auf die Fähigkeit zur Skalierbarkeit der Lösung hin.

Es gilt abschließend anzumerken, dass die Wiederverwendung von Elementen innerhalb der Infrastruktur bei den oben beschriebenen Versuchen keine Berücksichtigung findet. Die betrachteten Fälle bilden somit ein Worst-Case Szenario ab. Für jeden der Dienstaufrufe wird der vollständige Aufrufpfad durchlaufen und jeweils ein neuer MAA initiiert, um die Ergebnisse nicht zu verfälschen. Ziel innerhalb eines Wirkbetriebs ist hingegen die Wiederverwendung von MAA zur Reduktion der Performanzeinbußen.

5.6 VERWANDTE ARBEITEN

In diesem Abschnitt werden die zum AMAS.KOM Framework verwandten Arbeiten vorgestellt und von dem in dieser Arbeit vorgestellten Framework abgegrenzt. Grundsätzlich kann als Ergebnis der Untersuchung verwandter Arbeiten festgehalten werden, dass existierende Arbeiten nur Einzelaspekte der verteilten Überwachung und Steuerung von Diensten adressieren. Eine integrierte Betrachtung, speziell auch für das Anwendungsfeld dienstbasierter Architekturen, existiert zum aktuellen Zeitpunkt nicht. Ebenfalls existiert kein Ansatz, der eine umfangreiche Steuerung von Diensten zur Realisation von Gegenmaßnahmen zulässt.

Die im Folgenden vorgestellten Ansätze lassen sich anhand verschiedener Aspekte kategorisieren. Die Unterscheidung in die Überwachung *funktionaler oder nichtfunktionaler Eigenschaften* stellt ein erstes Unterscheidungsmerkmal dar. Ebenfalls kann in *zentral organisierte oder verteilte Ansätze* unterschieden werden. Darüber hinaus spielt das durch den jeweiligen Ansatz adressierte *Einsatzgebiet* eine Rolle. Ein Teil

	Überwachte Eigenschaften	Verteilung	Einsatzgebiet	Funktionsumfang
[SM06]	f	z	WF	Ü
[Rob05]	f+nf	z	WF	Ü
[BGG04]	f	z	WF	Ü
[BG05]	f+nf	z	WF	Ü
[BGP06]	f+nf	z	WF	Ü
[BGP08]	f+nf	z	WF	Ü
[MRD08]	f+nf	z	WF	Ü+S
[CPEV05]	nf	z	WF	Ü+S
[BGR+07]	nf	z	WF	Ü+S
[CCK+06]	nf	z	D	Ü
[LDK04]	nf	z	D	Ü
[LKD+03a]	nf	z	D	Ü
[SDS05]	f+nf	v	D	Ü+S
[ZHG99]	nf	v	NK	Ü
[GGGO99]	nf	v	NK	Ü
[LPK01]	nf	v	NK	Ü

Tabelle 16: Übersicht zu AMAS.KOM verwandter Arbeiten

der nachfolgend vorgestellten Ansätze wurde speziell für die Überwachung und teilweise auch die Steuerung von dienstbasierten Workflows entwickelt, wohingegen andere Ansätze aus dem Anwendungsfeld des klassischen Netzwerkmanagements stammen. Abschließend kann der *Funktionsumfang* der Ansätze als Unterscheidungsmerkmal herangezogen werden. Nicht alle Ansätze erlauben in Ergänzung zu der reinen Überwachung von Diensten die Steuerung der Dienste im Falle von Fehlern oder Abweichungen von Dienstgütevereinbarungen.

Der nachfolgende Abschnitt wird anhand der Unterscheidung in zentrale und verteilte Überwachung und Steuerung strukturiert. Darüber hinaus wird in einem gesonderten Abschnitt auf wichtige Standards und Spezifikationen eingegangen, die vielen der beschriebenen verwandten Arbeiten zu Grunde liegen. Eine Übersicht der im Folgenden diskutierten Ansätze sowie deren Einordnung in die eben beschriebene Kategorisierung wird in Tabelle 16 dargestellt. Die Charakterisierung der überwachten Eigenschaften erfolgt dabei in funktionale (f) und nicht-funktionale (nf) Eigenschaften sowie deren gleichzeitige Berücksichtigung (f+nf). Analog wird bei der Verteilung in zentrale (z) und verteilte (v) Ansätze unterschieden. Das Einsatzgebiet lässt sich in die Überwachung von Workflows (WF), einzelnen Diensten (D) oder Netzwerkkomponente (NK) unterscheiden. Der Funktionsumfang kann in reine Überwachung (Ü) sowie Überwachung und Steuerung (Ü+S) unterschieden werden. Eine Übersicht der Abkürzungen ist Gegenstand von Tabelle 17.

Abkürzung	Bedeutung
f	funktionale Eigenschaften
nf	nicht-funktionale Eigenschaften
z	zentraler Ansatz
v	verteilter Ansatz
WF	Workflows
D	Dienste
NK	Netzwerkkomponenten
Ü	Überwachung
S	Steuerung

Tabelle 17: Verwendete Abkürzungen

5.6.1 *Zentrale Ansätze*

Der erste der hier vorgestellten Ansätze zur Überwachung dienstbasierter Workflows stammt von Spanoudakis und Mahbub und verwendet Event Calculus zur Beschreibung funktionaler Anforderungen an eine Workflowausführung [SM06]. Die zentrale Überwachung eines in BPEL beschriebenen Geschäftsprozesses erfolgt parallel zu dessen Ausführung durch einen dezidierten Überwachungsprozess. Die Ausführung des eigentlichen Prozesses wird somit nicht beeinflusst, eine Anpassung dessen Prozessbeschreibung ist ebenfalls nicht notwendig.

Die Initiierung des Überwachungssystems erfolgt durch Politiken, die beschreiben, welche Dienste zu überwachen sind und welche Anforderungen es hierbei zu überprüfen gilt. Eine solche Politik, die zu überprüfende Muster von Ereignissen beinhaltet, wird aus der Analyse der Prozessbeschreibung durch die Transformation von formellen Anforderungen in Ereignismuster gewonnen. Die so gefundenen Ereignismuster werden später zur Auswertung der Ereignisdatenbank verwendet. Die Nutzung des Event Calculus zur Beschreibung der Ereignisse ermöglicht zusätzlich zur einfachen Mustererkennung auch die Folgerung über die vorhandenen Ereignisse zur Erzeugung neuen Wissens über das zu überwachende System.

Im weiteren Verlauf des Ansatzes sammelt eine Überwachungskomponente Ereignisse von der Process-Engine ein und legt diese zur späteren Auswertung in der Ergebnisdatenbank ab. Die Auswertung erfolgt erst lange nach dem Eintritt der Ereignisse. Sollten bei der Auswertung der Ergebnisdatenbank Abweichungen von den funktionalen Anforderungen durch den Vergleich mit den Ereignismustern festgestellt werden, so werden diese Abweichungen in einer gesonderten Datenbank dokumentiert. Diese Datenbank wird ihrerseits in festen Zeitintervallen für Berichtszwecke abgefragt.

Der Ansatz von Spanoudakis und Mahbub sieht keine Steuerung von Korrekturmaßnahmen im Falle von Anforderungsverletzungen vor. Die einzige Reaktion auf solche Anforderungsverletzungen besteht in deren Meldung an die aufrufende Instanz, welche in den meisten Fällen der Nutzer des Geschäftsprozesses ist.

Ein vergleichbarer Ansatz, der sich allerdings weniger direkt auf dienstbasierte Workflows als allgemein auf Informationssysteme in Unternehmen bezieht, ist das von Robinson in [Rob05] beschriebene *ReqMon* Framework. Das *ReqMon* Framework ermöglicht die Ableitung von zu überwachenden funktionalen und teilweise auch nicht-funktionalen Anforderungen aus allgemeinen Anforderungsbeschreibungen sowie deren Überführung in ausführbare Überwachungskomponenten. Grundlage hierfür ist der kombinierte Einsatz von Techniken der Anforderungsanalyse aus dem Bereich des Software Engineerings sowie Verfahren zur Überwachung der Ausführung von Softwaresystemen. Der Schwerpunkt des Ansatzes liegt auf der Durchgängigkeit der Betrachtung von einer wenig formalisierten Anforderung hin zu einer ausführbaren Überwachungskomponente.

Zur Beschreibung der Anforderungen kommt eine Form der temporalen Logik zum Einsatz. Die in dieser Form beschriebenen Anforderungen werden parallel zur Ausführung eines Prozesses bzw. des den Prozess ausführenden Systems durch eine zentrale Instanz überprüft.

Vergleichbar zum vorangegangenen Ansatz wird auch durch das Framework von Robinson ausschließlich die Meldung einer Abweichung an die beauftragende Instanz als Reaktion auf Abweichungen von Anforderung unterstützt. Die Steuerung evtl. notwendiger Korrekturmaßnahmen ist nicht Bestandteil des Verfahrens.

Baresi et al. beschreiben in ihren Arbeiten verschiedene Ansätze zur Überwachung funktionaler und nicht-funktionaler Anforderungen an Dienste und dienstbasierte Workflows, die nachfolgend im Überblick vorgestellt werden. In [BGG04] beschreiben die Autoren die Überwachung von funktionalen Anforderungen sowie die Überprüfung auf Ausführungsfehler und Timeouts zur Laufzeit eines Prozesses. Hierzu wird eine Prozessbeschreibung auf Basis von BPEL durch Annotationen in einer eigens hierfür entwickelten Sprache zur Beschreibung von Vor- und Nachbedingungen ausgezeichnet. Diese Annotationen werden als Kommentare innerhalb des BPEL-Codes integriert. Die Überwachung von Laufzeitfehlern und Timeouts kann dabei direkt in BPEL abgebildet werden, wohingegen die funktionalen Anforderungen durch wiederverwendbare Überwachungswebservices geprüft werden, welche ihrerseits in den BPEL-Prozess eingebunden sind.

In diesem Ansatz sind die Geschäftslogik, die Anforderungen sowie mögliche Maßnahmen zur Behandlung von Anforderungsverletzungen innerhalb derselben BPEL-Prozessbeschreibung spezifiziert, was die Wartbarkeit einer solchen Lösung bei sich ändernden Anforderungen negativ beeinflusst. Darüber hinaus machen die Autoren in diesem Ansatz keine Unterscheidung zwischen Fehlern in der Ausführung eines Prozesses oder Abweichungen von Anforderungen, die beispielsweise innerhalb einer Leistungsvereinbarung definiert sind. Aus technischer Sicht werden diese Abweichungen innerhalb des Prozesses entweder als reale Fehler behandelt oder aber einfach ignoriert. Das Hauptproblem der starken Verknüpfung von Geschäftslogik sowie Überwachungs- und Steuerungselementen wird in einer Erweiterung des Ansatzes durch Einführung von extern spezifizierten Anforderungsdefinitionen in Form von Politiken beseitigt [BG05]. Eine weitere Ergänzung des bestehenden Ansatzes erfolgt in [BGP06] durch die Einführung einer zu WS-Policy konformen Spezifikationssprache für Politiken sowie durch die Erweiterung des Ansatzes um Konzepte der aspektorientierten Programmierung (AOP) in [BGP08], welche an dieser Stelle nicht weiter behandelt werden.

Hauptunterscheidungsmerkmale zu dem in dieser Arbeit beschriebenen Ansatz sind die Unterstützung der Verteilung von Überwachung und Steuerung sowie die Wiederverwendbarkeit von MAU. Gleichermaßen ist die Unterstützung der Behandlung von Anforderungsabweichungen jeglicher Art ein Alleinstellungsmerkmal von AMAS.KOM im Vergleich zu den eben aufgeführten Arbeiten von Baresi et al.

Ein umfassender, zentraler Ansatz für die Überwachung und Steuerung von dienstbasierten Workflows, der die aspektorientierte Programmierung nutzt, ist das von Moser et al. beschriebene VieDAME (Vienna Dynamic Adaptation and Monitoring Environment for WSBPEL) [MRD08]. Die Anwendung von AOP ermöglicht die Erweiterung bestehender Softwaresysteme, in dem innerhalb einer existierender Software Punkte definiert werden können, an denen Veränderungen bzw. Erweiterungen (die so genannten Aspekte) der Funktionalität des Systems vorgenommen werden können. Hierzu muss die eigentliche Software in ihrer Funktion nicht verändert werden. Erweiterungen können ebenfalls zur Laufzeit an- und ausgeschaltet werden.

VieDAME erlaubt die Überwachung von BPEL-basierten Prozessen sowie den Austausch von Diensten zur Laufzeit, ohne dass hierbei der eigentliche BPEL-Prozess verändert werden muss. Im Detail kann für jeden verwendbaren Dienst definiert werden, ob dieser ausgetauscht werden kann und unter welchen Bedingungen dies geschehen soll. Entsprechende Regelsätze und Politiken sowie die alternativen Dienste werden innerhalb des Systems hinterlegt. Somit integriert das Verfahren die Überwachung von Diensten mit Ansätzen zur Dienstselektion sowie der Unterstützung sämtlicher zur Gewährleistung der Kompatibilität von auszutauschenden Diensten notwendigen Transformationsschritte.

Großer Nachteil des VieDAME-Ansatzes ist es, dass die BPEL-Engine für die Unterstützung des AOP-basierten Ansatzes angepasst werden muss, was die Verwendung des Ansatzes stark limitiert. Gleichermaßen führt die enge Verknüpfung des Überwachungsansatzes mit BPEL dazu, dass nur auf die von BPEL unterstützten Mechanismen zur Behandlung von Fehlern zurückgegriffen werden kann. Eine feingranulare Behandlung von Anforderungsverletzungen ist nicht möglich.

Die bisher vorgestellten Ansätze wurden als ausschließliche Überwachungs- und teilweise Steuerungsansätze konzipiert. Nachfolgend werden Ansätze und Verfahren vorgestellt, die Bestandteil einer größeren Lösung sind. Den Anfang bilden zwei Ansätze, die aus dem Bereich des integrierten Dienstgütemanagements von Diensten stammen. Hierbei bezeichnet das integrierte Dienstgütemanagement die Eigenschaft eines Systems neben der reinen Messung von Dienstgüteparametern, auch Dienste anhand von Dienstgüteparametern auszuwählen und häufig auch zu Workflows zu komponieren. Die Behandlung von Abweichungen ist in beiden Ansätzen in Grundzügen erkennbar, wobei der zur Verfügung stehende Funktionsumfang gering ist.

Der erste der exemplarisch für diese Kategorie vorgestellten Ansätze stammt von Canfora et al. [CPEV05]. Canfora et al. beschreiben ein Verfahren, auf dessen Basis es für die Ausführung von BPEL-Prozessen möglich ist, zur Laufzeit Dienste auszutauschen (dem so genannten *Replanning* einer Komposition von Diensten), sofern sie die an sie gestellten Anforderungen nicht erfüllen können. Hierzu macht sich der Ansatz ein Proxy-Konzept zum Abfangen von Dienstaufrufen zu Nutze, welches ähnlich zu dem in dieser Arbeit verwendeten Konzept ist. Während die Prozessbeschreibung zunächst nur abstrakte Dienste zueinander in Beziehung setzt, ist es

durch Nutzung des Proxy möglich, konkrete Dienste zur Laufzeit zuzuordnen und auszuführen. Die Proxy-Komponente kann zur Auswahl auf eine Sammlung funktional gleicher Dienste zurückgreifen. Zur Laufzeit des Systems kann anschließend auf Basis von Überwachungsdaten entschieden werden, ob ein selektierter Dienst problematisch ist und ausgetauscht werden muss. Die Überwachung ist in Form von in die Prozessbeschreibung integrierten Messpunkten abgebildet, welche beim Erreichen Überwachungsdienste aufrufen, die ihrerseits die geloggten Daten in ein Monitoring-Repository überführen.

Ebenfalls ein vollständiges System zum Management von Dienstgüte stellt Berbner et al. mit WSQoSX vor [BGR+07]. WSQoSX ermöglicht die Auswahl und Komposition von Diensten anhand von Dienstgütemerkmalen, die auch durch Überwachungsinformationen gewonnen werden können. Dienste können auf dieser Basis beim Systemstart zur Berücksichtigung innerhalb eines Workflows ausgewählt, aber auch zur Laufzeit bei Verletzung von Dienstgüteanforderungen ausgetauscht werden (vgl. dem Ansatz von Canfora et al.). Die Überwachung unterschiedlicher nichtfunktionaler Merkmale ist integraler Bestandteil von WSQoSX, wobei die eigentliche Gewinnung der Messdaten nicht Gegenstand der Arbeit ist. Für Messungen wird auf einen externen Messdienst zurückgegriffen.

Als nächste Vertreter aus der Klasse der zentralen Ansätze sollen zwei Ansätze von Ludwig et al. vorgestellt werden, welche in der Literatur eine große Verbreitung und eine hohe Akzeptanz besitzen. Die in [LDK04] vorgestellte CREMONA (Creation and Monitoring of Agreements) Architektur beschreibt eine Middleware, mit deren Hilfe Verhandlungen zwischen Dienstnutzer und -anbieter automatisiert abgewickelt werden können. Hierbei kommt die WS-Agreement-Sprache zum Einsatz, welche Bestandteile von WS-Policy beinhaltet, durch die Anforderungen der an einer Geschäftsbeziehung beteiligten Parteien beschrieben werden. Durch den wiederholten Austausch sowie die Anpassung dieser Beschreibungen wird versucht, einen Konsens zwischen den Partnern zu erzeugen. Nachdem dieser Konsens erreicht wurde, gilt es die Vereinbarung laufend auf ihre Einhaltung zu überwachen. Hierfür werden auf Seiten der Architektur entsprechende Komponenten (den so genannten *Compliance Monitors*) definiert, deren Arbeitsweise allerdings nicht weiter ausgeführt wird.

Ähnlich verhält es sich mit der im Rahmen der WSLA-Spezifikation (siehe Abschnitt 2.3.3) eingeführten abstrakten Basisarchitektur derselben Autoren [LKD+03a]. Zum Zwecke der Überwachung der durch WSLA beschriebenen Dienstgüteanforderungen an Webservices führen die Autoren das Konzept des *Runtime Management* ein, welches grundsätzlich drei Kernbestandteile unterscheidet. Die *Measurement* Komponente empfängt die Messdaten aus dem unterliegenden Überwachungssystem und ermittelt auf Basis der WSLA-Beschreibung zugehörige Metriken für die Dienstausführung. Diese Ergebnisse werden im Rahmen der *Condition Evaluation* im Hinblick auf die Einhaltung der Anforderungen innerhalb des WSLA interpretiert. Wird eine Verletzung einer Anforderung festgestellt, so wird ein Ereignis an die eigentliche *Management* Komponente gesendet, die für die Verarbeitung des Events zuständig ist. Die Komponente *Management* ist in der WSLA-Spezifikation bewusst abstrakt gehalten – eine mögliche Handhabung von Abweichungen bzw. Verletzungen wird nicht ausgeführt.

Beide Ansätze ermöglichen grundsätzlich die Berücksichtigung von funktionalen und nicht-funktionalen Anforderungen. Sie definieren Grundlagen einer Vielzahl der vorangegangenen Arbeiten, lassen aber im Hinblick auf die Ausführung der eigentlichen Überwachung und Steuerung eine Vielzahl von Fragestellungen ungeklärt.

Natürlich existieren zu dieser Arbeit verwandte Arbeiten nicht ausschließlich im Forschungsumfeld. Abschließend soll an dieser Stelle exemplarisch ein kommerziell verfügbares Produkt vorgestellt werden, welches einen Teil der erwünschten Funktionalitäten anbietet.

Oracle bietet mit dem Web Services Manager ein Tool zur zentralisierten Überwachung und der Umsetzung von Politiken für Webservices an, welches zur Datengewinnung auf verteilte Überwachungseinheiten zurückgreift [CCK+06]. Entsprechende Überwachungseinheiten können autarke Komponenten innerhalb einer Infrastruktur sein oder aber innerhalb eines Applikationsservers integriert werden. Der Web Services Manager überwacht hierbei einfache Dienstgüteaspekte, wie z. B. die Antwortzeiten eines Dienstes oder die Einhaltung von Sicherheitsrichtlinien. Als mögliche Reaktionen auf Abweichungen von Anforderungen unterstützt das Tool Berichtsfunktionen sowie die Selektion von alternativen Politiken.

5.6.2 *Verteilte Ansätze*

Für den Bereich der verteilten Ansätze existieren gleichermaßen eine Reihe von Arbeiten, von denen eine Auswahl im Folgenden vorgestellt wird.

Ein Ansatz, der die Verteilung einzelner weniger Überwachungsdienste innerhalb einer Infrastruktur nativ unterstützt und nicht nur auf einer verteilten Datenerhebung beruht, ist das Verfahren von Schmietendorf et al. [SDS05]. Der Schwerpunkt des Ansatzes liegt in der Überwachung von nicht-funktionalen Eigenschaften einzelner Dienste, die durch unabhängige dritte Parteien innerhalb einer Infrastruktur durchgeführt wird. Zielsetzung ist hierbei die Verbesserung der Reputation einzelner Anbieter und deren Dienste durch die unabhängige Bewertung deren Qualität. Die Überwachung zur Laufzeit und im Wirkbetrieb steht nicht im Fokus des Ansatzes, vielmehr wird auf ein gesondert durchgeführtes Testen von Diensten sowie Vorhersagen des zukünftigen Verhaltens zurückgegriffen. Eine Kooperation der einzelnen Überwachungsdienste ist hierbei nicht vorgesehen. Der Ansatz ist ebenfalls ein reiner Reporting-Ansatz, d. h. die Behandlung von Abweichungen zuvor spezifizierter Anforderungen wird nicht unterstützt.

Eine Reihe von Ansätzen aus dem dieser Arbeit verwandten Themenbereich des Netzwerkmanagements greift zur Realisierung von Überwachungsfunktionalitäten auf den Einsatz von Agententechnologie zurück. Die Anwendung von Softwareagenten ermöglicht den Ansätzen eine einfache Verteilung innerhalb der zu überwachenden Infrastrukturen.

Ein früher Vertreter dieser Kategorie ist das *NetDoctor* Framework von Zapf et al. [ZHG99]. Die Autoren beschreiben darin einen Ansatz, mit dessen Hilfe dezentral durch Verwendung mobiler Softwareagenten Überwachungsinformationen gewonnen und auf Basis der Informationen Korrekturmaßnahmen an Netzkomponenten

vorgenommen werden können. Die so genannten *Health Agents* sind dabei mobil, d. h. sie können innerhalb der Infrastruktur jederzeit neu platziert werden. Sie integrieren darüber hinaus die in der Infrastruktur bestehenden Netzwerkmanagementverfahren auf Basis des Simple Network Management Protocol (SNMP). Grundüberlegung des Ansatzes ist hierbei, analog zu einer der Grundlagen der vorliegenden Arbeit, dass die Minimierung des Abstands zwischen der Überwachungs- und Steuerungseinheit sowie der zu überwachenden Ressource einen positiven Einfluss auf die Auslastung des Gesamtsystems besitzt. Dieser Einfluss wird durch quantitative Überlegungen nachgewiesen.

Der nächste hier vorgestellte agentenbasierte Ansatz für das Netzwerkmanagement stammt von Gavalas et al. [GGGO99] und adressiert die im Vergleich zu klassischen Netzwerkmanagementverfahren effiziente Akquise von Überwachungsinformationen innerhalb zu überwachenden Infrastrukturen. Weitere Zielsetzung der Autoren ist eine hohe Skalierbarkeit sowie Robustheit des Überwachungsverfahrens. Im Detail beinhaltet der Ansatz verschiedene Polling-Verfahren zur Sammlung von Überwachungsinformationen sowohl für die Verarbeitung in Echtzeit als auch für die nachgelagerte Verarbeitung. Im Rahmen des Pollings bewegen sich Agenten zur Sammlung der benötigten Informationen innerhalb der Infrastruktur bzw. einzelnen Partitionen dieser Infrastruktur von Netzkomponente zu Netzkomponente, um dabei die benötigten Informationen aufzunehmen. Durch die Partitionierung des zu überwachenden Bereichs sowie die Zuweisung von Agenten, welche diese Partitionen überwachen, lässt sich die Komplexität der gesamten Überwachung reduzieren. Das Verfahren macht sich hierbei die Mobilität der Agenten zu Nutze. Der Ansatz berücksichtigt dabei keine Behandlung von Fehlern oder Abweichung von Anforderungen.

Den Gedanken der Partitionierung der zur überwachenden Infrastruktur greift auch die Arbeit von Liotta et al. auf [LPK01]. Der in ihr beschriebene Ansatz beschäftigt sich mit der Platzierung von Agenten innerhalb einer Infrastruktur und somit implizit auch um deren Partitionierung. Hierzu entwickeln die Autoren einen einfachen, verteilten Algorithmus zur Bestimmung der Lokation der Überwachungseinheiten, welcher darauf basiert, dass sich in der Infrastruktur bewegende Agenten nach und nach die für die Bestimmung der Lokalität notwendigen Topologieinformationen ermitteln. Bei Ausfall einzelner Verbindung innerhalb der Topologie erlaubt das Verfahren eine Relokation der Überwachungseinheiten.

5.6.3 *Standards und Spezifikationen*

Im Umfeld der Standards existieren ebenfalls Ansätze, die als verwandte Arbeiten sowie auch als Grundlagen für die vorliegende Arbeit betrachtet werden können.

Den Anfang macht hierbei das Simple Network Management Protocol, einem Standard, welcher als Inhalt verschiedener so genannter Requests for Comments (RFC) durch die Internet Engineering Task Force (IETF) verabschiedet wurde.[27] SNMP

27 Version 1 (1988) ist in RFC 1155 bis 1157, Version 2 (1996) in RFC 1901, 1905 und 1906 sowie Version 3 (2002) in RFC 3410 bis 3418 beschrieben. Alle sind parallel im Einsatz, die weiteste Verbreitung besitzt Version 2.

dient zur Überwachung und Fernkonfiguration von Netzwerkkomponenten, welche durch eine zentrale Instanz durchgeführt wird [Sta99]. Zwar sind SNMP-basierte Systeme die Basis jedes zentral aufgebauten Überwachungssystems, die Informationsgewinnung an sich erfolgt aber in einer verteilten Form. SNMP ist weit verbreitet – die meisten Netzwerkkomponenten, wie beispielsweise Router oder Switches, besitzen entsprechende Schnittstellen, über welche unter Einsatz des SNMP auf die Komponente zugegriffen werden kann. Kernelement von SNMP-kompatiblen Netzwerkkomponenten sind hierbei Agenten, die auf einer Netzwerkkomponente die Datengewinnung übernehmen und über SNMP dem Managementsystem zur Verfügung stellen. SNMP definiert das zum Austausch von Überwachungs- und Steuerungsinformationen notwendige Datenformat, allerdings nicht die Eigenschaften, die auf den unterschiedlichen Netzwerkkomponenten überwacht werden können. Dies wird durch die Management Information Base (MIB) realisiert, einer hierarchischen Klassifizierung möglicher durch SNMP zu überwachender und zu steuernder Eigenschaften einer Netzwerkkomponente. Durch in den MIB spezifizierten Managed Objects wird herstellerübergreifend eindeutig definiert, welche Funktion durch die Komponente angeboten wird bzw. welche Eigenschaften überwacht werden können.

SNMP stellt eine Ergänzung zur Gewinnung von Informationen für die Entscheidungsfindung und Ermittlung von Verteilungen in AMAS.KOM dar.

Ebenfalls ein Ansatz aus dem Bereich des Netzwerkmanagements ist der Remote Monitoring (RMON) Standard, welcher in seiner letzten Version in den RFC 2819 und RFC 4502 beschrieben ist (vgl. hierzu auch [Sta99] und [GGGO99]). RMON stellt eine Erweiterung der MIB aus dem SNMP Standard dar, welche es ermöglicht, die periodischen Abfragen von Überwachungsinformationen der SNMP Agenten zur Entlastung des Netzwerks zu reduzieren. Dies geschieht durch die verteilte Sammlung und Speicherung von Überwachungsinformationen innerhalb der durch einen Agenten überwachten Domäne. Zielsetzung ist die Gewinnung umfangreicher Informationen über Netzsegmente, die proaktiv zur Vermeidung von Fehlern oder Leistungsengpässen herangezogen werden können. Auf Grund der hohen Ressourcenanforderungen von RMON-fähigen Agenten ist die Verbreitung von RMON auf Netzwerkkomponenten gering.

Die Idee von RMON ist ebenfalls mit AMAS.KOM verwandt. Die dezentrale Sammlung von Informationen sowie die Unterstützung von dezentral initiierten Korrekturmaßnahmen ist, wenn auch auf Applikationsschicht, gleichermaßen in dem in dieser Arbeit vorgestellten Ansatz berücksichtigt. Die geringe Verbreitung entsprechender Netzwerkkomponenten lässt RMON im Bereich des Netzwerkmanagements SMTP unterlegen sein.

Abschließend soll der von OASIS spezifizierte WSDM-Standard vorgestellt und diskutiert werden. WSDM liegt aktuell in Version 1.1 vor. Der Standard besteht aus zwei Kernkonzepten, welche als *Management using Web Services* (MUWS – vgl. [BV06a] und [BV06b]) und *Management of Web Services* (MOWS – vgl. [WS06]) bezeichnet und in getrennten Standardisierungsdokumenten spezifiziert werden. Die Zielsetzung des WSDM-Standards ist der von SNMP vergleichbar. WSDM definiert ein Protokoll und geeignete Mechanismen, um Dienste und Ressourcen verschiedenster Art innerhalb eines Netzwerks zu überwachen und zu steuern. MUWS beschreibt den Zugriff auf Ressourcen innerhalb eines Netzwerks durch die Nutzung

einer Webservice-Schnittstelle. Zur Realisierung eines solchen einheitlichen Zugriffs wird durch den Standard eine Reihe von Managementfunktionen sowie grundlegender Metriken definiert, die eine Ressource in jedem Fall anbieten muss, um zu WSDM konform zu sein. Dies impliziert, dass ein mit MUWS überwachter Dienst eine bestimmte Schnittstelle implementieren muss, so dass eine Überarbeitung seiner Codebasis bzw. eine geeignete Instrumentierung notwendig wird, auch wenn diese Schnittstelle selbst nicht durch den Standard definiert wird. MOWS definiert hingegen, wie Webservices selbst als Ressourcen zu verwalten sind. Hierunter fallen z. B. die Beschreibung von Webservice-Funktionalitäten oder die Eigenschaften eines Webservices. MOWS nutzt MUWS zur Überwachung und Steuerung von Ressourcen.

WSDM kann, analog zu SNMP, als Komplement zu AMAS.KOM gesehen werden. WSDM macht keine Aussagen über eine mögliche Ausgestaltung des Überwachungs- und Steuerungssystems. Eine Einbindung der durch WSDM bereit gestellten Funktionen in AMAS.KOM ist wünschenswert und wird bereits in Form erster Plugins unterstützt.

5.7 ZUSAMMENFASSUNG

Im fünften Kapitel dieser Arbeit wird das AMAS.KOM Framework beschrieben, welches einen Ansatz zur Realisierung einer verteilten Überwachung und Steuerung von Diensten darstellt. AMAS.KOM besteht im Kern aus einem Vorgehensmodell mit integriertem Transformationsprozess, mit dessen Hilfe eine nicht überwachte Workflow-Instanz in eine überwachte Workflow-Instanz überführt werden kann, sowie aus einem Architekturentwurf auf Basis der Kombination der Paradigmen dienstbasierter Architekturen und agentenbasierter Systeme.

Als Ergänzung der konzeptionellen Betrachtung des Frameworks wird gleichermaßen ein Vorschlag zur technischen Umsetzung des Frameworks präsentiert. Zur technischen Umsetzung kommen aktuelle Webservice-Technologien zum Einsatz, wie z. B. SOAP, WSDL und WSBPEL. Diese werden innerhalb einer auf dem FIPA-Standard für Softwareagenten basierenden Agentenplattform für die Überwachung und Steuerung von Diensten zum Einsatz gebracht. Die Implementierung dient als Proof-of-Concept für das AMAS.KOM Framework in existierenden Webservice-Infrastrukturen.

Abschließend erfolgt die Untersuchung der Implementierung des Frameworks auf Basis der zuvor selektierten Technologien im Hinblick auf mögliche Performanzveränderungen durch das Einfügen von AMAS.KOM in eine bestehende dienstbasierte Infrastruktur. Als Ergebnis der Untersuchung kann festgehalten werden, dass die Antwortzeit der Einzeldienste innerhalb eines durch AMAS.KOM überwachten und gesteuerten Systems sich im Durchschnitt um rund 14 % verschlechtert. Der Durchsatz reduziert sich im gleichen Fall ebenfalls um rund 12 %. In beiden Fällen wird die Wiederverwendung von Überwachungs- und Steuerungseinheiten nicht berücksichtigt, was die Höhe des gemessenen Einflusses von AMAS.KOM auf die Gesamtperformanz der Dienste begründet.

6 ZUSAMMENFASSUNG UND AUSBLICK

In diesem Kapitel werden die wesentlichen Ergebnisse der Arbeit zusammengefasst sowie ein Ausblick auf mögliche Anknüpfungspunkte für zukünftige Forschungsarbeiten im Umfeld der Überwachung und Steuerung von Diensten in dienstbasierten Architekturen gegeben.

6.1 ZUSAMMENFASSUNG

Die Umsetzung dienstbasierter Architekturen und dienstbasierter Workflows innerhalb eines Unternehmens bringt eine Reihe von Herausforderungen mit sich, welche sowohl durch die Fach- als auch die IT-Abteilungen zu adressieren sind. Besondere Bedeutung besitzt in betrieblichen Anwendungsfeldern die Gewährleistung hoher Dienstgüte von Geschäftsprozessen, Workflows sowie der unterliegenden IT-Systeme, welche durch Einführung eines geeigneten Dienstgütemanagements erreicht werden kann.

Im Rahmen des Qualitätsmanagements von Diensten und dienstbasierten Workflows gilt es in einem ersten Schritt Anforderungen an die jeweiligen Geschäftsprozesse zu definieren sowie nachfolgend die daraus resultierenden Anforderungen an die unterliegenden IT-Systeme abzuleiten. Allerdings reicht eine reine Spezifikation und vertragliche Fixierung der Anforderungen nicht aus – vielmehr müssen zur Laufzeit die Einhaltung der Anforderungen überwacht und im Abweichungsfall geeignete Gegenmaßnahmen initiiert werden können.

Von besonderer Relevanz bei der Überwachung und Steuerung von Diensten als Teil dienstbasierter Workflows ist die Berücksichtigung der zum Teil komplexen, unternehmensübergreifenden Geschäftsbeziehungen zwischen Dienstnutzern und Dienstanbietern. Hieraus resultiert, dass ein Dienstgütemanagementsystem häufig nicht auf alle zur optimalen Überwachung und Steuerung notwendigen Informationen zentral zugreifen kann, da diese nur in fremden Kontrollsphären gewonnen werden können. Darüber hinaus haben sich die im SOA-Umfeld weit verbreiteten zentral organisierten Dienstgütemanagementsysteme in Untersuchungen als Leistungsengpass bei der Detektion und Behandlung von Anforderungsverletzungen herausgestellt.

Die vorliegende Arbeit schlägt zur Adressierung dieser Herausforderungen die Verteilung von Überwachungs- und Steuerungseinheiten innerhalb dienstbasierter Infrastrukturen vor und liefert zu deren Realisierung mehrere Beiträge.

Der Fokus der Arbeit liegt hierbei auf der Entwicklung und Evaluierung von Strategien zur effizienten Verteilung der Überwachungs- und Steuerungseinheiten in dienstbasierten Infrastrukturen. Die im Rahmen der Arbeit entwickelten Strategien vereinfachen die bisher komplexe und rechenintensive Ermittlung von Verteilungen, sodass diese durch ressourcenbeschränkte, in der Infrastruktur verteilte und

autonom agierende Überwachungs- und Steuerungseinheiten selbstständig ermittelt werden können.

Die im Rahmen der Arbeit entwickelten Strategien arbeiten mit Informationen aus vorhandenen Leistungsvereinbarungen, in welchen die Dienstnutzer und -anbieter die Ziele der Leistungserbringung definieren, sowie mit Messdaten, die zur Laufzeit gewonnen werden. Mit Hilfe der Verteilungsstrategien wird eine kostenminimierende Verteilung innerhalb der Infrastruktur ermittelt sowie diese bedarfsweise angepasst.

Die Ergebnisse der Arbeit zeigen, wie sich das NP-schwere Verteilungsproblem für Überwachungs- und Steuerungseinheiten geeignet modellieren und durch exakte sowie heuristische Verfahren lösen lässt. Die Evaluation des Verteilungsansatzes sowie der damit verbundenen exakten und heuristischen Lösungsverfahren werden in einer eigens hierfür entwickelten Simulationsumgebung durchgeführt. Die Ergebnisse der Simulation zeigen, dass in allen untersuchten Szenarien bei Betrachtung der maximalen untersuchten Topologiegröße durch die Verteilung von Überwachungs- und Steuerungseinheiten Kosteneinsparungen zwischen 11 % und 54 % im Vergleich zur zentralen Überwachung und Steuerung erreicht werden können. Die von dem vorgeschlagenen heuristischen Verbesserungsverfahren erreichte Lösungsgüte, d. h. dessen Annäherung an die optimale Lösung, erreicht zwischen 85 % und 99 %, wobei dessen Laufzeitverhalten in Bezug auf die Topologiegröße linear ist. Die relative Laufzeit, d. h. die Laufzeit der Heuristik im Verhältnis zu der Laufzeit des optimalen Verfahrens, beträgt in diesen Fällen zwischen 8 % und 17 %. Auf Grund dieser Ergebnisse eignen sich sowohl das Verbesserungsverfahren als auch das unwesentlich schlechtere Eröffnungsverfahren sehr gut für den Einsatz in zeitkritischen und ressourcenbeschränkten Szenarien, wie den zuvor beschriebenen ressourcenbeschränkten, autonom agierenden Überwachungs- und Steuerungseinheiten.

Die Untersuchung des Verhältnisses von Topologiegröße, Anzahl von Dienstanbietern sowie der Anzahl der zu platzierenden Einheiten zeigt, dass je nach Szenario und eingesetztem Lösungsverfahren 20 % bis 70 % der Anzahl der Dienstanbieter als Einheiten zu platzieren sind, um eine maximal kostenreduzierende verteilte Überwachung und Steuerung zu gewährleisten. Eine solche Abschätzung für ein gegebenes Szenario macht zukünftig die Anwendung von noch effizienteren Lösungsverfahren möglich, die eine vordefinierte Anzahl von Einheiten in Polynomialzeit verteilt.

Weitere Beiträge der Arbeit liegen im Bereich der Entwicklung und prototypischen Umsetzung von Hilfsmitteln für die verteilte Überwachung und Steuerung, insbesondere der Adressierung dabei auftretender methodischer, architektonischer und technologischer Herausforderungen.

Die Arbeit untersucht, wie die verteilte Überwachung und Steuerung innerhalb bestehender dienstbasierter Systeme integriert werden kann. Das in dieser Arbeit entwickelte AMAS.KOM Framework stellt hierfür einen Architekturentwurf zur Verfügung, welcher auf der Anwendung des Paradigmas dienstbasierter Architekturen basiert. Dessen Umsetzung auf Basis der Kombination von Webservice-Technologien mit einer Plattform für Softwareagenten erlaubt die Realisierung autonomer und verteilbarer Überwachungs- und Steuerungseinheiten. Weiterer Baustein des Frameworks ist dessen integriertes Vorgehensmodell zur Erzeugung überwachter dienstbasierter Workflows. Auf Grundlage eines Transformationsprozesses kann eine Workflowdefinition mit zugehöriger Spezifikation von Anforderungen und Reaktionen

auf Abweichungen teilautomatisiert in eine überwachte Workflowinstanz überführt und zur kontrollierten Ausführung gebracht werden.

Eine Grundlage für die Verteilung von Überwachungs- und Steuerungslogik ist die gleichzeitige Modellierung von Anforderungen und Reaktionen auf Abweichungen, welche den Arbeitsauftrag und Handlungsspielraum einer Einheit spezifiziert. Aus Mangel an einer geeigneten Spezifikationssprache wird in dieser Arbeit die WS-Re2Policy-Sprache entwickelt und in AMAS.KOM integriert.

Als Nachweis für die Realisierbarkeit der in dieser Arbeit vorgestellten Ansätze wird AMAS.KOM prototypisch unter Verwendung der Webservice-Technologie implementiert und im Hinblick auf durch die Integration einer zusätzlichen Verteilungsschicht entstehenden Performanzveränderungen hin untersucht. Die Untersuchungen zeigen, dass im schlechtesten Fall, d. h. im Fall ohne Wiederverwendung von Überwachungs- und Steuerungseinheiten, mit einem Rückgang des Durchsatzes von rund 12 % sowie einer Verschlechterung der Antwortzeiten von rund 14 % zu rechnen ist. Beide Veränderungen sind im Hinblick auf den Nutzen der Gesamtlösung vernachlässigbar.

6.2 AUSBLICK

Die Überwachung und Steuerung von Diensten und dienstbasierten Workflows stellt einen Aspekt des Dienstgütemanagements in dienstbasierten Architekturen dar. Im nachfolgenden Abschnitt werden ausgewählte Forschungsfelder betrachtet, die einerseits die vorliegende Arbeit ergänzen und die andererseits zu erschließende Anwendungsfelder für Inhalte dieser Arbeit darstellen.

Eine erste Übersicht über ergänzende Forschungsthemen innerhalb der Anwendungsdomäne der dienstbasierten Workflows liefert die Forschungsagenda aus Abbildung 1 in Kapitel 1. Während die Bereiche der grundlegenden Funktionalitäten sowie der komponierten Dienste mittlerweile die Schwerpunkte zahlreicher laufender Arbeiten darstellen, so herrscht im Bereich der verwalteten Dienste weiterhin erhöhter Forschungsbedarf. Gemeinsamer Nenner möglicher Arbeiten in diesem Bereich sind die Reduktion und Bewältigung von Komplexität, die durch die Integration von Diensten unterschiedlicher, unternehmensexterner Anbieter in die Abläufe eines Unternehmens entsteht.

Ein Schritt in diese Richtung bietet die Selbstmanagementfähigkeit von Systemen, die in den letzten Jahren in viele Bereiche der Informationsverarbeitung Einzug gehalten hat (vgl. [HM08]). Prinzipien wie das durch IBM geprägte *Autonomic Computing* beinhalten Konzepte wie die Selbstkonfiguration von Systemen, die selbstständig durchgeführte Überwachung und Steuerung der Ablaufe, die Korrektur von Fehlern und anderen Abweichungen zur Wiederherstellung eines ordnungsgemäßen Systemzustands sowie das selbstständige Reagieren auf Sicherheitsrisiken. Ziel ist neben der Komplexitätsreduktion im Betrieb auch hier die Verbesserung der Robustheit der Gesamtsysteme. In diesem Bereich stellen die Arbeiten von Miede et al. einen direkten Anknüpfungspunkt an die vorliegende Arbeit dar, in welchen Kooperationsmechanismen zwischen teilautonom handelnden Überwachungs- und Steuerungseinheiten auf Basis des AMAS.KOM Frameworks untersucht werden (vgl. [MBR+09] und [MBP+09]). Kern der Arbeiten ist die arbeitsteilige Überwachung und Steuerung eines Dienstes durch Cluster von Softwareagenten, welche auf Grund

unvollständigen lokalen Wissens auf die Kooperation mit anderen Agenten angewiesen sind. Schulte et al. knüpfen ebenfalls direkt an die Ergebnisse dieser Arbeit an, indem sie in [SRSS09] semantische Erweiterungsmöglichkeiten für WS-Re2Policy untersuchen.

Weiterhin von großer Relevanz für den Bereich der dienstbasierten Workflows ist das automatisierte Verhandeln von Dienstgütevereinbarungen und anderen vertraglichen Parametern. Gerade in Szenarien wie dem Internet der Dienste, in welchem Dienste vergleichbar heutiger Internetressourcen zur Nutzung zur Verfügung stehen, spielt die Automatisierung solcher Verhandlungen in dynamischen Geschäftsbeziehungen eine wichtige Rolle. Hierbei kommt der Berücksichtigung des Vertrauens der Verhandlungspartner untereinander und der Reputation von Dienstanbietern ein hoher Stellenwert zu (vgl. z. B. [KHEP08] und [HLW08]). Auch in diesem Forschungsfeld können Überwachungs- und Steuerungssysteme Einsatz finden, wobei die in dieser Arbeit vorgestellten Ansätze entsprechend angepasst werden müssten.

Gleichermaßen ein hochrelevantes Forschungsfeld im Bereich dienstbasierter Architekturen und Workflows ist die Überwachung und Steuerung der gesamten IT-Organisation zur Sicherstellung des Erreichens der Unternehmensziele durch den Einsatz von IT im Allgemeinen und SOA im Speziellen. IT- und SOA-Governance stellen hierfür geeignete Konzepte zur Verfügung, welche aus Sicht der Organisation und des Managements eines Unternehmens den gesamten Lebenszyklus von IT bzw. Diensten und dienstbasierten Architekturen zu verwalten versuchen (vgl. [NERS08] und [RMS08]). Die Governance definiert beispielsweise Richtlinien, Empfehlungen und Steuerungsinstrumente für das Management von Anforderungen, das Architekturmanagement, die Entwicklung oder den Zukauf von Diensten oder die Überwachung der Einhaltung der Anforderungen an diese Dienste. Weiterhin nimmt die Sicherstellung der Übereinstimmung mit rechtlichen Rahmenbedingungen (Compliance) einen zunehmend größeren Stellenwert ein. Die Beiträge zur Überwachung und Steuerung von Diensten aus dieser Arbeit stellen hierfür einen Baustein dar.

Zum Abschluss dieser Arbeit werden ausgewählte Anwendungsfelder dienstbasierter Architekturen sowie der Überwachung und Steuerung von Diensten präsentiert, welche abseits des Bereichs der dienstbasierten Workflows zukünftig hohes Forschungspotential versprechen.

Bisherige SOA-Ansätze der Hersteller bis hin zum Internet der Dienste gehen zumeist davon aus, dass Dienstnutzer mit wechselnden Dienstanbietern interagieren. Gleichermaßen Berücksichtigung finden sollte allerdings auch der sich ändernde Dienstnutzer, welcher beispielsweise in mobilen Szenarien Dienste komponiert und konsumiert. Erste Forschungsvorhaben, wie z. B. die Projekte *SERVICES to go!* und *GREEN Mobility*, wurden vor Kurzem im Umfeld des vom Bundesministeriums für Wirtschaft und Technologie geförderten Forschungsvorhaben THESEUS TEXO[28] gestartet, um die Verbindung von Diensten und Dienstmanagement, mobiler Kommunikationstechnologie sowie die Integration mit Kontextinformationen in Anwendungsszenarien zu untersuchen.

Ebenfalls lässt sich das SOA-Paradigma auf andere Diensttypen anwenden, welche sich in der Granularität von den in dieser Arbeit zu Grunde gelegten Diensten unterscheiden. Prinzipiell kann die Dienstabstraktion auf Sensoren in Sensornetzwerken

28 http://www.theseus-programm.de/ – abgerufen am 05.04.2009.

(vgl. [SKR+08]) gleichermaßen angewandt werden, wie auf die Komponenten eines Systems zur Heimautomatisierung (vgl. *embeddedSOA* bei [SBS+09]). In allen Fällen müssen Dienste beschrieben, gefunden, komponiert und zur Laufzeit überwacht und gesteuert werden. Lediglich die Umsetzungstechnologien sowie der durch die Dienste bereitgestellte Funktionsumfang unterscheiden sich merklich von bisherigen Einsatzszenarien dienstbasierter Architekturen.

[AAD+07a] Agrawal, Ashish ; Amend, Mike ; Das, Manoj ; Ford, Mark ; Keller, Chris ; Kloppmann, Matthias ; König, Dieter ; Leymann, Frank ; Müller, Ralf ; Pfau, Gerhard ; Plösser, Karsten ; Rangaswamy, Ravi ; Rickayzen, Alan ; Rowley, Michael ; Schmidt, Patrick ; Trickovic, Ivana ; Yiu, Alex ; Zeller, Matthias: *WS-BPEL Extension for People (BPEL4People) Version 1.0*. 2007 http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel4people/BPEL4People_v1.pdf – abgerufen am 05.04.2009.

[AAD+07b] Agrawal, Ashish ; Amend, Mike ; Das, Manoj ; Ford, Mark ; Keller, Chris ; Kloppmann, Matthias ; König, Dieter ; Leymann, Frank ; Müller, Ralf ; Pfau, Gerhard ; Plösser, Karsten ; Rangaswamy, Ravi ; Rickayzen, Alan ; Rowley, Michael ; Schmidt, Patrick ; Trickovic, Ivana ; Yiu, Alex ; Zeller, Matthias: *Web Services Human Task (WS-HumanTask) Version 1.0*. 2007 http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel4people/WS-HumanTask_v1.pdf – abgerufen am 05.04.2009.

[ABC+02] Ackermann, Jörg ; Brinkop, Frank ; Conrad, Stefan ; Fettke, Peter ; Frick, Andreas ; Glistau, Elke ; Jaekel, Holger ; Kotlar, Otto ; Loos, Peter ; Mrech, Heike ; Ortner, Erich ; Overhage, Sven ; Raape, Ulrich ; Sahm, Stephan ; Schmietendorf, Andreas ; Teschke, Thorsten ; Turowski, Klaus: *Vereinheitlichte Spezifikation von Fachkomponenten*. Gesellschaft für Informatik, Arbeitskreis 5.10.3, 2002 http://www.fachkomponenten.de – abgerufen am 05.04.2009.

[ABFG04] Austin, Daniel ; Barbir, Abbie ; Ferris, Christopher ; Garg, Sharad: *Web Services Architecture Requirements*. W3C Working Group Note, 2004 (W3C Working Group Note). http://www.w3.org/TR/wsa-reqs/ – abgerufen am 05.04.2009.

[ACD+07] Andrieux, Alain ; Czajkowski, Karl ; Dan, Asit ; Keahey, Kate ; Ludwig, Heiko ; Nakata, Toshiyuki ; Pruyne, Jim ; Rofrano, John ; Tuecke, Steve ; Xu, Ming: *Web Services Agreement Specification (WS-Agreement)*. OGF Grid Resource Allocation Agreement Protocol Working Group, 2007 http://www.ogf.org/documents/GFD.107.pdf – abgerufen am 05.04.2009.

[ACKM04] Alonso, Gustavo ; Casati, Fabio ; Kuno, Harumi ; Machiraju, Vijay ; Carey, M. J. (Hrsg.) ; Ceri, S. (Hrsg.): *Web Services: Concepts, Architectures and Applications*. Springer-Verlag, 2004

[ACLL07] Agrawal, Dakshi ; Calo, Seraphin ; Lee, Kang-Won ; Lobo, Jorge: Issues in Designing a Policy Language for Distributed Management of IT Infrastructures. In: *Tagungsband 10th IFIP/IEEE International Symposium on Integrated Network Management*, 2007, S. 30–39

[AFM+05] Akkiraju, Rama ; Farrell, Joel ; Miller, John ; Nagarajan, Meenakshi ; Schmidt, Marc-Thomas ; Sheth, Amit ; Verma, Kunal: *Web Service Semantics (WSDL-S)*. W3C, 2005 (W3C Member Submission). http://www.w3.org/Submission/WSDL-S/ – abgerufen am 05.04.2009.

[AH02] Aalst, Wil van d. ; Hee, Kees V.: *Workflow Management: Models, Methods, and Systems*. MIT Press, 2002

[AHA+00] Alonso, Gustavo ; Hagen, Claus ; Agrawal, Divyakant ; Abbadi, Amr E. ; Mohan, C.: Enhancing the Fault Tolerance of Workflow Management Systems. In: *IEEE Concurrency* 8 (2000), Nr. 3, S. 74–81

[AJB99] Albert, Reka ; Jeong, Hawoong ; Barabasi, Albert-Laszlo: Internet: Diameter of the World-Wide Web. In: *Nature* 401 (1999), Nr. 6749, S. 130–131

[BA99] Barabasi, Albert-Laszlo ; Albert, Reka: Emergence of scaling in random networks. In: *Science* 286 (1999), S. 509–512

[BB03a] Balke, Wolf-Tilo ; Badii, Atta: Assessing Web services quality for call-by-call outsourcing. In: *Tagungsband 4th International Conference on Web Information Systems Engineering Workshop*, 2003, S. 173–181

[BB03b] Barabasi, Albert-Laszlo ; Bonabeau, Eric: Scale-free networks. In: *Scientific American* 288 (2003), Mai, Nr. 5, S. 60–69

[BBC+98] Blake, Steven ; Black, David ; Carlson, Mark ; Davies, Elwyn ; Wang, Zheng ; Weiss, Walter: *An Architecture for Differentiated Services*. IETF Network Working Group, 1998 http://tools.ietf.org/html/rfc2475/ – abgerufen am 05.04.2009.

[BBC+06] Bajaj, Siddharth ; Box, Don ; Chappell, Dave ; Curbera, Francisco ; Daniels, Glen ; Hallam-Baker, Phillip ; Hondo, Maryann ; Kaler, Chris ; Langworthy, Dave ; Nadalin, Anthony ; Nagaratnam, Nataraj ; Prafullchandra, Hemma ; Riegen, Claus von ; Roth, Daniel ; Schlimmer, Jeffrey ; Sharp, Chris ; Shewchuk, John ; Vedamuthu, Asir ; Yalçinalp Ümit ; Orchard, David: *Web Services Policy 1.2 – Framework (WS-Policy)*. W3C, 2006 (W3C Recommendation). http://www.w3.org/Submission/WS-Policy/ – abgerufen am 05.04.2009.

[BBF+06] Bieberstein, Norbert ; Bose, Sanjay ; Fiammante, Marc ; Jones, Keith ; Shah, Rawn: *Service-Oriented Architecture (SOA) Compass: Business Value, Planning, and Enterprise Roadmap*. IBM Press, 2006

[BCCT05] Brambilla, Marco ; Ceri, Stefano ; Comai, Sara ; Tziviskou, Christina: Exception handling in workflow-driven Web applications. In: *Tagungsband 14th International Conference on World Wide Web*, 2005, S. 170–179

[BCG07] Bellifemine, Fabio ; Caire, Giovanni ; Greenwood, Dominic: *Developing multi-agent systems with JADE*. John Wiley & Sons, 2007

[BCS94] Braden, Robert ; Clark, David ; Shenker, Scott: *Integrated Services in the Internet Architecture: an Overview*. IETF Network Working Group, 1994 http://www.ietf.org/rfc/rfc1633.txt – abgerufen am 05.04.2009.

[Ber08] Berbner, Rainer: *Dienstgüteunterstützung für Service-orientierte Workflows*. Books on Demand, 2008

[BF08] Beck, Roman ; Franke, Jochen: Designing Reputation and Trust Management Systems. In: *Journal of Electronic Commerce in Organizations* 6 (2008), Nr. 4, S. 8–29

[BFT04] Bayer, Thomas ; Frotscher, Thilo ; Teufel, Marc ; Wang, Dapeng (Hrsg.): *Java Web Services mit Apache Axis*. Entwickler.Press, 2004

[BG05] Baresi, Luciano ; Guinea, Sam: Towards Dynamic Monitoring of WS-BPEL Processes. In: *Tagungsband 3rd International Conference on Service Oriented Computing*, 2005, S. 269–282

[BGG04] Baresi, Luciano ; Ghezzi, Carlo ; Guinea, Sam: Smart monitors for composed services. In: *Tagungsband 2nd International Conference on Service Oriented Computing*, 2004, S. 193–202

[BGP06] Baresi, Luciano ; Guinea, Sam ; Plebani, Pierluigi: WS-Policy for Service Monitoring. In: *Tagungsband 6th Workshop Technologies for E-Services*, 2006, S. 72–83

[BGP08] Baresi, Luciano ; Guinea, Sam ; Pasquale, Liliana: Towards a Unified Framework for the Monitoring and Recovery of BPEL Processes. In: *Tagungsband Workshop on Testing, Analysis, and Verification of Web Services and Applications*, 2008, S. 15–19

[BGR+05] Berbner, Rainer ; Grollius, Tobias ; Repp, Nicolas ; Heckmann, Oliver ; Ortner, Erich ; Steinmetz, Ralf: An approach for the Management of Service-oriented Architecture (SOA) based Application Systems. In: *Tagungsband Workshop Enterprise Modelling and Information Systems Architectures*, 2005, S. 208–221

[BGR+07] Berbner, Rainer ; Grollius, Tobias ; Repp, Nicolas ; Eckert, Julian ; Heckmann, Oliver ; Ortner, Erich ; Steinmetz, Ralf: Management of Service-oriented Architecture (SOA)-based Application Systems. In: *Enterprise Modelling and Information Systems Architectures* 1 (2007), Nr. 2, S. 14–26

[BHK+03] Box, Don ; Hondo, Maryann ; Kaler, Chris ; Maruyama, Hiroshi ; Nadalin, Anthony ; Nagaratnam, Nataraj ; Patrick, Paul ; Riegen, Claus von ; Shewchuk, John: *Web Services Policy Assertions Language (WS-PolicyAssertions)*. 2003

[BHL+04] Burstein, Mark ; Hobbs, Jerry ; Lassila, Ora ; Martin, David ; McDermott, Drew ; McIlraith, Sheila ; Narayanan, Srini ; Paolucci, Massimo ; Parsia, Bijan ; Payne, Terry ; Sirin, Evren ; Srinivasan, Naveen ; Sycara, Katia: *OWL-S: Semantic Markup for Web Services*. W3C Web-Ontology Working Group, 2004 (W3C Member Submission). `http://www.w3.org/Submission/OWL-S/` – abgerufen am 05.04.2009.

[BHM+04] Booth, David ; Haas, Hugo ; McCabe, Francis ; Newcomer, Eric ; Champion, Michael ; Ferris, Chris ; Orchard, David: *Web Services Architecture*. W3C Web Services Architecture Working Group, 2004 `http://www.w3.org/TR/ws-arch/` – abgerufen am 05.04.2009.

[BHMS05] Berbner, Rainer ; Heckmann, Oliver ; Mauthe, Andreas ; Steinmetz, Ralf: Eine Dienstgüte unterstützende Web-Service-Architektur für flexible Geschäftsprozesse. In: *Wirtschaftsinformatik* 47 (2005), September, Nr. 4, S. 268–277

[Bir05] Birmann, Kenneth P.: *Reliable Distributed Systems: Technologies, Web Services, and Applications*. Springer-Verlag, 2005

[BL06] Bichler, Martin ; Lin, Kwei-Jay: Service-Oriented Computing. In: *IEEE Computer* 39 (2006), März, Nr. 3, S. 99–101

[BL07] Booth, David ; Liu, Canyang K.: *Web Services Description Language (WSDL) Version 2.0 Part 0: Primer*. W3C Web Services Description Working Group, 2007 (W3C Recommendation). `http://www.w3.org/TR/wsdl20-primer/` – abgerufen am 05.04.2009.

[BM99] Borgida, Alex ; Murata, Takahiro: Tolerating exceptions in workflows: a unified framework for data and processes. In: *Tagungsband International Joint Conference on Work Activities Coordination and Collaboration*, 1999, S. 59–68

[BMS04] Berbner, Rainer ; Mauthe, Andreas ; Steinmetz, Ralf: Unterstützung dynamischer E-Finance-Geschäftsprozesse. In: *Tagungsband der Konferenz Elektronische Geschäftsprozesse*, 2004, S. 44–54

[BRL07] Baligand, Fabien ; Rivierre, Nicolas ; Ledoux, Thomas: A Declarative Approach for QoS-Aware Web Service Compositions. In: *Tagungsband 5th International Conference on Service-Oriented Computing*, 2007, S. 422–428

[BSR+06a] Berbner, Rainer ; Spahn, Michael ; Repp, Nicolas ; Heckmann, Oliver ; Steinmetz, Ralf: An Approach for Replanning of Web Service Workflows. In: *Tagungsband 12th Americas Conference on Information Systems*, 2006

[BSR+06b] Berbner, Rainer ; Spahn, Michael ; Repp, Nicolas ; Heckmann, Oliver ; Steinmetz, Ralf: Heuristics for QoS-aware Web Service Composition. In: *Tagungsband 4th IEEE International Conference on Web Services*, 2006, S. 72–79

[Buc08] Buchmann, Johannes: *Einführung in die Kryptographie*. Springer-Verlag, 2008

[Bun08] Bundesamt für Sicherheit in der Informationstechnik: *SOA-Security-Kompendium: Sicherheit in Service-orientierten Architekturen*. Bundesamt für Sicherheit in der Informationstechnik, 2008 http://www.bsi.bund.de/literat/studien/soa/index.htm – abgerufen am 05.04.2009.

[BV06a] Bullard, Vaughn (Hrsg.) ; Vambenepe, William (Hrsg.): *Web Services Distributed Management: Management Using Web Services (MUWS 1.1) – Part 1*. OASIS Web Services Distributed Management TC, 2006 http://docs.oasis-open.org/wsdm/wsdm-muws1-1.1-spec-os-01.pdf – abgerufen am 05.04.2009.

[BV06b] Bullard, Vaughn (Hrsg.) ; Vambenepe, William (Hrsg.): *Web Services Distributed Management: Management Using Web Services (MUWS 1.1) – Part 2*. OASIS Web Services Distributed Management TC, 2006 http://docs.oasis-open.org/wsdm/wsdm-muws2-1.1-spec-os-01.pdf – abgerufen am 05.04.2009.

[BWK05] Beck, Roman ; Wigand, Rolf T. ; König, Wolfgang: The Diffusion and Efficient Use of Electronic Commerce among Small and Medium-sized Enterprises: An International Three-Industry Survey. In: *Electronic Markets* 15 (2005), Nr. 1, S. 38–52

[BWW03] Bullinger, Hans-Jörg ; Warnecke, Hans-Jürgen ; Westkämper, Engelbert: *Neue Organisationsformen im Unternehmen: Ein Handbuch für das moderne Management*. Springer-Verlag, 2003

[BZW98] Brenner, Walter ; Zarnekow, Rüdiger ; Wittig, Hartmut: *Intelligente Softwareagenten – Grundlagen und Anwendung*. Springer-Verlag, 1998

[CBZ09] Comes, Diana ; Bleul, Steffen ; Zapf, Michael: Management of Business Processes with the BPRules Language in Service Oriented Computing. In: *Tagungsband Workshops der Wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen*, 2009, S. 1–12

[CCG+02] Chen, Qian ; Chang, Hyunseok ; Govindan, Ramesh ; Jamin, Sugih ; Shenker, Scott J. ; Willinger, Walter: The origin of power laws in Internet topologies revisited. In: *Tagungsband 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, 2002, S. 608–617

[CCK+06] Chu, Ken ; Cordero, Orlando ; Korf, Mario ; Pickersgill, Catherine ; Whitmore, Robin: *Oracle SOA Suite Developer's Guide*. Oracle, 2006

[CD99] Clark, James ; DeRose, Steve: *XML Path Language (XPath) – Version 1.0*. W3C XSL Working Group and XML Linking Working Group, 1999 (W3C Recommendation). http://www.w3.org/TR/xpath – abgerufen am 05.04.2009.

[Cer02] Cerami, Ethan: *Web Services Essentials: Distributed Applications with XML-RPC, SOAP, UDDI and WSDL*. O'Reilly Media, 2002

[CFL+05] Chaudet, Claude ; Fleury, Eric ; Lassous, Isabelle G. ; Rivano, Hervé ; Voge, Marie-Emilie: Optimal positioning of active and passive monitoring devices. In: *Tagungsband 2005 ACM Conference on Emerging Network Experiment and Technology*, 2005, S. 71–82

[CHRR04] Clement, Luc (Hrsg.) ; Hately, Andrew (Hrsg.) ; Riegen, Claus von (Hrsg.) ; Rogers, Tony (Hrsg.): *UDDI Version 3.0.2*. OASIS UDDI Spec Technical Committee, 2004 (Draft). `http://uddi.org/pubs/uddi_v3.htm` – abgerufen am 05.04.2009.

[CIB+06] Cantieni, Gion R. ; Iannaccone, Gianluca ; Barakat, Chadi ; Diot, Christophe ; Thiran, Patrick: Reformulating the monitor placement problem: optimal network-wide sampling. In: *Tagungsband ACM CoNEXT Conference*, 2006, S. 1–12

[CMRW07] Chinnici, Roberto ; Moreau, Jean-Jacques ; Ryman, Arthur ; Weerawarana, Sanjiva: *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. W3C Web Services Description Working Group, 2007 (W3C Recommendation). `http://www.w3.org/TR/wsdl20/` – abgerufen am 05.04.2009.

[CPEV05] Canfora, Gerardo ; Penta, Massimiliano D. ; Esposito, Raffaele ; Villani, Maria L.: QoS-Aware Replanning of Composite Web Services. In: *Tagungsband IEEE International Conference on Web Services*, 2005, S. 121–129

[CPEV08] Canfora, Gerardo ; Penta, Massimiliano D. ; Esposito, Raffaele ; Villani, Maria L.: A framework for QoS-aware binding and re-binding of composite web services. In: *Journal of Systems and Software* 81 (2008), Nr. 10, S. 1754–1769

[CS03] Choi, Sumi ; Shavitt, Yuval: Proxy Location Problems and Their Generalizations. In: *Tagungsband 23rd International Conference on Distributed Computing Systems*, 2003, S. 898–904

[CS04] Chudak, Fabián A. ; Shmoys, David B.: Improved Approximation Algorithms for the Uncapacitated Facility Location Problem. In: *SIAM Journal on Computing* 33 (2004), Nr. 1, S. 1–25

[CSM+04] Cardoso, Jorge ; Sheth, Amit P. ; Miller, John A. ; Arnold, Jonathan ; Kochut, Krys J.: Modeling Quality of Service for Workflows and Web Service Processes. In: *Web Semantics Journal* 1 (2004), Nr. 3, S. 281–308

[DD95] Domschke, Wolfgang ; Drexl, Andreas: *Logistik: Standorte*. Oldenbourg Verlag, 1995

[DD98] Domschke, Wolfgang ; Drexl, Andreas: *Einführung in Operations Research*. Springer-Verlag, 1998

[DDK+04] Dan, Asit ; Davis, Doug ; Kearney, Robert ; Keller, Alexander ; King, Richard P. ; Kuebler, Dietmar ; Ludwig, Heiko ; Polan, Mike ; Spreitzer, Mike ; Youssef, Alaa: Web services on demand: WSLA-driven automated management. In: *IBM Systems Journal* 43 (2004), Nr. 1, S. 136–158

[DK97] Domschke, Wolfgang ; Krispin, Gabriela: Location and layout planning: A survey. In: *OR Spektrum* 19 (1997), S. 181–194

[DPTS06] Denaro, G. ; Pezzé, M. ; Tosi, D. ; Schilling, Daniela: Towards self-adaptive service-oriented architectures. In: *Tagungsband Workshop on Testing, Analysis, and Verification of Web Services and Applications*, 2006, S. 10–16

[Eck07] Eckert, Claudia: *IT-Sicherheit: Konzepte – Verfahren – Protokolle*. Oldenbourg Wissenschaftlicher Verlag, 2007

[EEM+08] Eckert, Julian ; Ertogrul, Deniz ; Miede, André ; Repp, Nicolas ; Steinmetz, Ralf: Resource Planning Heuristics for Service-oriented Workflows. In: *Tagungsband 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 2008, S. 591–597

[EHH+08] Engels, Gregor ; Hess, Andreas ; Humm, Bernhard ; Juwig, Oliver ; Lohmann, Marc ; Richter, Jan-Peter: *Quasar Enterprise: Anwendungslandschaften serviceorientiert gestalten*. dpunkt.verlag, 2008

[EK04] Ellis, Avy ; Kauferstein, Michael: *Dienstleistungsmanagement: Erfolgreicher Einsatz von prozessorientiertem Service Level Management*. Springer-Verlag, 2004

[EPR+07] Eckert, Julian ; Pandit, Krishna ; Repp, Nicolas ; Berbner, Rainer ; Steinmetz, Ralf: Worst-Case Performance Analysis of Web Service Workflows. In: *Tagungsband 9th International Conference on Information Integration and Web-based Application & Services*, 2007, S. 67–77

[Erl05] Erl, Thomas: *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, 2005

[ERS+07] Eckert, Julian ; Repp, Nicolas ; Schulte, Stefan ; Berbner, Rainer ; Steinmetz, Ralf: An Approach for Capacity Planning for Web Service Workflows. In: *Tagungsband 13th Americas Conference on Information Systems*, 2007

[ESN+08] Eckert, Julian ; Schulte, Stefan ; Niemann, Michael ; Repp, Nicolas ; Steinmetz, Ralf: Worst-Case Workflow Performance Optimization. In: *Tagungsband 3rd International Conference on Internet and Web Applications and Services*, 2008, S. 632–637

[ESR+08] Eckert, Julian ; Schulte, Stefan ; Repp, Nicolas ; Berbner, Rainer ; Steinmetz, Ralf: Queuing-based Capacity Planning Approach for Web Service Workflows Using Optimization Algorithms. In: *Tagungsband IEEE International Conference on Digital Ecosystems and Technologies*, 2008, S. 313–318

[FGM+99] Fielding, Roy T. ; Gettys, James ; Mogul, Jeffrey C. ; Nielsen, Henrik F. ; Masinter, Larry ; Leach, P. ; Berners-Lee, T.: *RFC 2616 – Hypertext Transfer Protocol (HTTP/1.1)*. IETF Network Working Group, 1999 `http://www.ietf.org/rfc/rfc2616.txt` – abgerufen am 05.04.2009.

[Fie00] Fielding, Roy T.: *Architectural Styles and the Design of Network-based Software Architectures*, University of California, Irvine, Dissertation, 2000

[FJP+01] Francis, Paul ; Jamin, Sugih ; Paxson, Vern ; Zhang, Lixia ; Gryniewicz, Daniel ; Jin, Yixin: IDMaps: a global internet host distance estimation service. In: *IEEE/ACM Transactions on Networking* 9 (2001), Oktober, Nr. 5, S. 525–540

[FRT99] Fellner, Klement J. ; Rautenstrauch, Claus ; Turowski, Klaus: Fachkomponenten zur Gestaltung betrieblicher Anwendungssysteme. In: *IM Information Management & Consulting* 14 (1999), Nr. 2, S. 25–34

[FT00] Fielding, Roy T. ; Taylor, Richard N.: Principled design of the modern Web architecture. In: *Tagungsband 22nd International Conference on Software Engineering*, 2000, S. 407–416

[FT02] Fielding, Roy T. ; Taylor, Richard N.: Principled design of the modern Web architecture. In: *ACM Transactions on Internet Technology* 2 (2002), Nr. 2, S. 115–150

[GC07] Government Commerce, Office of: *ITIL Version 3: Service Operation*. The Stationery Office, 2007

[GGGO99] Gavalas, Damianos ; Greenwood, Dominic ; Ghanbari, Mohammed ; O'Mahony, Mike: Using Mobile Agents for Distributed Network Performance Management. In: *Tagungsband 3rd International Workshop on Intelligent Agents for Telecommunication Applications*, 1999, S. 96–112

[GHM+07a] Gudgin, Martin ; Hadley, Marc ; Mendelsohn, Noah ; Moreau, Jean-Jacques ; Nielsen, Henrik F. ; Karmarkar, Anish ; Lafon, Yves: *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. W3C XML Protocol Working Group, 2007 (W3C Recommendation). `http://www.w3.org/TR/soap12-part1/` – abgerufen am 05.04.2009.

[GHM+07b] Gudgin, Martin ; Hadley, Marc ; Mendelsohn, Noah ; Moreau, Jean-Jacques ; Nielsen, Henrik F. ; Karmarkar, Anish ; Lafon, Yves: *SOAP Version 1.2 Part 2: Adjuncts (Second Edition)*. W3C XML Protocol Working Group, 2007 (W3C Recommendation). `http://www.w3.org/TR/soap12-part2/` – abgerufen am 05.04.2009.

[Had08] Haddadi, Hamed: *Topological Characteristics of IP Networks*, University College London, Dissertation, November 2008

[HB04] Haas, Hugo (Hrsg.) ; Brown, Allen (Hrsg.): *Web Services Glossary*. W3C Web Services Architecture Working Group, 2004 `http://www.w3.org/TR/ws-gloss/` – abgerufen am 14.04.2009

[HB09] Hafner, Michael ; Breu, Ruth: *Security Engineering for Service-Oriented Architectures*. Springer-Verlag, 2009

[Hec06] Heckmann, Oliver: *The Competitive Internet Service Provider: Network Architecture, Interconnection, Traffic Engineering and Network Design*. John Wiley & Sons, 2006

[HJ06] Humm, Bernhard ; Juwig, Oliver: Eine Normalform für Services. In: Biel, Bettina (Hrsg.) ; Book, Matthias (Hrsg.) ; Gruhn, Volker (Hrsg.): *Tagungsband Konferenz Software Engineering*, 2006, S. 99–110

[HLJ04] Hung, Patrick C. K. ; Li, Haifei ; Jeng, Jun-Jang: WS-Negotiation: An Overview of Research Issues. In: *Tagungsband 37th Annual Hawaii International Conference on System Sciences*, 2004, S. 10033.2

[HLO03] Horton, Joseph D. ; López-Ortiz, Alejandro: On the number of distributed measurement points for network tomography. In: *Tagungsband 3rd ACM SIGCOMM Conference on Internet Measurement*, 2003, S. 204–209

[HLW08] Hudert, Sebastian ; Ludwig, Heiko ; Wirtz, Guido: Negotiating Service Levels – A Generic Negotiation Framework for WS Agreement. In: *Tagungsband 20th International Conference on Software Engineering & Knowledge Engineering*, 2008, S. 587–592

[HM08] Huebscher, Markus C. ; McCann, Julie A.: A survey of autonomic computing – degrees, models, and applications. In: *ACM Computing Surveys* 40 (2008), Nr. 3, S. 1–28

[Hol95] Hollingsworth, David: *The Workflow Reference Model*. Workflow Management Coalition, 1995

[HPSS03] Heckmann, Oliver ; Piringer, Michael ; Schmitt, Jens ; Steinmetz, Ralf: On realistic network topologies for simulation. In: *Tagungsband ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research*, 2003, S. 28–32

[HSL+06] Heckmann, Oliver ; Steinmetz, Ralf ; Liebau, Nicolas ; Buchmann, Alejandro ; Eckert, Claudia ; Kangasharju, Jussi ; Mühlhäuser, Max ; Schürr, Andreas: Qualitätsmerkmale von Peer-to-Peer-Systemen / Technische Universität Darmstadt. 2006 (KOM-TR-2006-03) – Technical Report

[Hum08] Humm, Bernhard: Was ist eigentlich ein Service? In: *Softwaretechnik-Trends* 28 (2008), November, Nr. 4, S. 8–11

[IT97] ITU-T: *X.641: Information technology - Quality of Service: Framework*. International Telecommunication Union, 1997 (Data Networks and Open System Communication)

[JBS97] Jablonski, Stefan ; Böhm, Markus ; Schulze, Wolfgang: *Workflow-Management*. dpunkt.verlag, 1997

[JE07] Jordan, Diane (Hrsg.) ; Evdemon, John (Hrsg.): *Web Services Business Process Execution Language Version 2.0*. OASIS Web Services Business Process Execution Language Technical Committee, 2007 `http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html` – abgerufen am 05.04.2009.

[JMS02b] Jin, Li-Jie ; Machiraju, Vijay ; Sahai, Akhil: Analysis on Service Level Agreement of Web Services / HP Laboratories Palo Alto. Version: Juni 2002. `http://www.hpl.hp.com/techreports/2002/HPL-2002-180.pdf`. HP Laboratories, Palo Alto, Juni 2002 (HPL-2002-180) – Technical Report – abgerufen am 05.04.2009.

[Jos08] Josuttis, Nicolai: *SOA in der Praxis: System-Design für verteilte Geschäftsprozesse*. dpunkt.verlag, 2008

[JRS08] Janiesch, Christian ; Ruggaber, Rainer ; Sure, York: Eine Infrastruktur für das Internet der Dienste. In: *HMD – Praxis der Wirtschaftsinformatik* 261 (2008), Juni, Nr. 45, S. 71–79

[KBR+05] Kavantzas, Nickolas ; Burdett, David ; Ritzinger, Gregory ; Fletcher, Tony ; Lafon, Yves ; Barreto, Charlton: *Web Services Choreography Description Language Version 1.0*. W3C Web Services Choreography Working Group, 2005 (W3C Candidate Recommendation). `http://www.w3.org/TR/ws-cdl-10/` – abgerufen am 05.04.2009.

[KBS05] Krafzig, Dirk ; Banke, Karl ; Slama, Dirk: *Enterprise SOA: Service-oriented Architecture Best Practices*. Prentice Hall, 2005

[KC08] Kanneganti, Ramarao ; Chodavarapu, Prasad: *SOA Security*. Manning Publications, 2008

[KGT06] Knöpfel, Andreas ; Gröne, Bernhard ; Tabeling, Peter: *Fundamental Modeling Concepts: Effective Communication of IT Systems*. Wiley, 2006

[KGV83] KIRKPATRICK, Scott ; GELATT, Carl D. ; VECCHI, Mario P.: Optimization by Simulated Annealing. In: *Science* 220 (1983), Mai, Nr. 4598, S. 671–680

[KHEP08] KÖNIG, Stefan ; HUDERT, Sebastian ; EYMANN, Thorsten ; PAOLUCCI, Mario: Towards Reputation Enhanced Electronic Negotiations for Service Oriented Computing. In: *Tagungsband TRUST*, 2008, S. 273–291

[KKL03] KALEPU, Sravanthi ; KRISHNASWAMY, Shonali ; LOKE, Seng W.: Verity: a QoS metric for selecting Web services and providers. In: *Tagungsband 4th International Conference on Web Information Systems Engineering Workshops*, 2003, S. 131–139

[KL03] KELLER, Alexander ; LUDWIG, Heiko: The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. In: *Journal of Network and Systems Management* 11 (2003), März, Nr. 1, S. 57–81

[KPRR03] KAPPEL, Gerti (Hrsg.) ; PRÖLL, Birgit (Hrsg.) ; REICH, Siegfried (Hrsg.) ; RETSCHITZEGGER, Werner (Hrsg.): *Web Engineering: Systematische Entwicklung von Web-Anwendungen*. dpunkt.verlag, 2003

[LDK04] LUDWIG, Heiko ; DAN, Asit ; KEARNEY, Robert: Cremona: An Architecture and Library for Creation and Monitoring of WS-Agreements. In: *Tagungsband 2nd International Conference on Service Oriented Computing*, 2004, S. 65–74

[Ley01] LEYMANN, Frank: *Web Service Flow Language (WSFL 1.0)*. IBM Cooperation, 2001 http://xml.coverpages.org/WSFL-Guide-200110.pdf – abgerufen am 05.04.2009.

[Lie08] LIEBAU, Nicolas C.: *Trusted Accounting in Peer-to-Peer Environments – A Novel Token-based Accounting Scheme for Autonomous Distributed Systems*, Technische Universität Darmstadt, Dissertation, November 2008

[LJL+03] LEE, Kang C. ; JEON, Jong H. ; LEE, Won S. ; JEONG, Seong-Ho ; PARK, Sang-Won: *QoS for Web Services: Requirements and Possible Approaches*. Korea : W3C Working Group, 2003 http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/ – abgerufen am 05.04.2009.

[LKD+03a] LUDWIG, Heiko ; KELLER, Alexander ; DAN, Asit ; KING, Richard P. ; FRANCK, Richard: *Web Service Level Agreement (WSLA) Language Specification*. IBM Cooperation, 2003 http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf – abgerufen am 05.04.2009.

[LKD+03b] LUDWIG, Heiko ; KELLER, Alexander ; DAN, Asit ; P.KING, Richard ; FRANCK, Richard: A Service Level Agreement Language for Dynamic Electronic Services. In: *Journal of Electronic Commerce Research* 3 (2003), März, Nr. 1 2, S. 43–59 – Kluwer Academic Publishers

[LMC+04] LININGTON, Peter F. ; MILOSEVIC, Zoran ; COLE, James B. ; GIBSON, Simon ; KULKARNI, Sachin ; NEAL, Stephen W.: A unified behavioural model and a contract language for extended enterprise. In: *Data and Knowledge Engineering* 51 (2004), Nr. 1, S. 5–29

[LN04] LOCKEMANN, Peter C. ; NIMIS, Jens: Flexibility through Multiagent Systems: Solution or Illusion? In: *Tagungsband International Conference on Current Trends in Theory and Practice of Computer Science*, 2004, S. 211–224

[LN05] LOCKEMANN, Peter C. ; NIMIS, Jens: Softwareagenten: Von den Eigenschaften zur Architektur. In: *it – Information Technology* 47 (2005), Januar, Nr. 1, S. 28–35

[LN06] Lockemann, Peter C. ; Nimis, Jens: Agent dependability as an architectural issue. In: *Tagungsband 5th International Joint Conference on Autonomous Agents and Multiagent Systems*, 2006, S. 1101–1103

[Loc03] Lockemann, Peter C.: Information system architectures: From art to science. In: *Tagungsband Fachtagung Datenbanksysteme für Business, Technologie und Web*, 2003, S. 1–27

[LPK01] Liotta, Antonio ; Pavlou, George ; Knight, Graham: A Self-adaptable Agent System for Efficient Information Gathering. In: *Tagungsband 3rd International Workshop on Mobile Agents for Telecommunication Applications*, 2001, S. 139–152

[LR00] Leymann, Frank ; Roller, Dieter: *Production Work Flow: Concepts and Techniques*. Prentice Hall, 2000

[LYCL05] Liu, Xianghui ; Yin, Jianping ; Cai, Zhiping ; Lv, Shaohe: On the Placement of Active Monitor in IP Network. In: *Tagungsband 3rd International Conference of Networking and Mobile Computing*, 2005, S. 1181–1187

[MBP+09] Miede, André ; Behuet, Jean-Baptiste ; Papageorgiou, Apostolos ; Niemann, Michael ; Repp, Nicolas ; Steinmetz, Ralf: Qualitative and Quantitative Aspects of Cooperation Mechanisms for Monitoring in Service-oriented Architectures. In: *Tagungsband 3rd IEEE International Conference on Digital Ecosystems and Technologies*, 2009

[MBR+09] Miede, André ; Behuet, Jean-Baptiste ; Repp, Nicolas ; Eckert, Julian ; Steinmetz, Ralf: Cooperation Mechanisms for Monitoring Agents in Service-oriented Architectures. In: *Tagungsband der 9. internationalen Tagung Wirtschaftsinformatik*, 2009, S. 749–758

[McG03] McGregor, Carolyn: A method to extend BPEL4WS to enable business performance measurement / University of Western Sydney - School of Computing & IT. 2003 (CIT/15/2003) – Technical Report

[Men02] Menascé, Daniel A.: QoS Issues in Web Services. In: *IEEE Internet Computing* 6 (2002), Nr. 6, S. 72–75

[MES+08] Melzer, Ingo ; Eberhard, Sebastian ; Sauter, Patrick ; Hilliger, Alexander ; Flehmig, Marcus ; Zengler, Barbara ; Dostal, Wolfgang ; Tröger, Peter ; Stumm, Boris ; Lipp, Matthias ; Jeckle, Mario ; Melzer, Ingo (Hrsg.): *Serviceorientierte Architekturen mit Web Services: Konzepte - Standards - Praxis*. Spektrum Akademischer Verlag, 2008

[ML07] Mitra, Nilo ; Lafon, Yves: *SOAP Version 1.2 Part 0: Primer (Second Edition)*. W3C XML Protocol Working Group, 2007 (W3C Recommendation). `http://www.w3.org/TR/2007/REC-soap12-part0-20070427/` – abgerufen am 05.04.2009.

[MLH+06] McCann, Julie A. ; Lemos, Rogerio D. ; Huebscher, Markus ; Rana, Omer F. ; Wombacher, Andreas: Can Self-managed systems be trusted?: Some views and trends. In: *Knowledge Engineering Review* 21 (2006), Nr. 3, S. 239–248

[MLM+06] MacKenzie, C. M. ; Laskey, Ken ; McCabe, Francis ; Brown, Peter F. ; Metz, Rebekah: *Reference Model for Service Oriented Architecture 1.0*. Organization for the Advancement of Structured Information Standards, 2006 `http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf` – abgerufen am 05.04.2009.

[MMB00] MEDINA, Alberto ; MATTA, Ibrahim ; BYERS, John: On the Origin of Power Laws in Internet Topologies. In: *ACM Computer Communication Review* 30 (2000), April, Nr. 2, S. 2000

[MMBBD02] MARILLY, Emmanuel ; MARTINOT, Olivier ; BETGÉ-BREZETZ, Stéphane ; DELÈGUE, Gérard: Requirements for service level agreement management. In: *Tagungsband IEEE Workshop on IP Operations and Management*, 2002, S. 57–62

[MMP06] MODAFFERI, Stefano ; MUSSI, Enrico ; PERNICI, Barbara: SH-BPEL: a self-healing plug-in for WS-BPEL engines. In: *Tagungsband 1st Workshop on Middleware for Service Oriented Computing*, 2006, S. 48–53

[Mog06] MOGUL, Jeffrey C.: Emergent (Mis)behavior vs. Complex Software Systems. In: *Tagungsband EuroSys*, 2006, S. 293–304

[MRD08] MOSER, Oliver ; ROSENBERG, Florian ; DUSTDAR, Schahram: Non-intrusive monitoring and service adaptation for WSBPEL. In: *Tagungsband 17th International Conference on World Wide Web*, 2008, S. 815–824

[MRR+53] METROPOLIS, Nicholas ; ROSENBLUTH, Arianna W. ; ROSENBLUTH, Marshall N. ; TELLER, Augusta H. ; TELLER, Edward: Equation of State Calculations by Fast Computing Machines. In: *Journal of Chemical Physics* 21 (1953), Nr. 6, S. 1087–1092

[MS02a] MAXIMILIEN, E. M. ; SINGH, Munindar P.: Conceptual model of web service reputation. In: *Special Interest Group on Management of Data Record* 31 (2002), Nr. 4, S. 36–41

[MS02b] MAXIMILIEN, E. M. ; SINGH, Munindar P.: Reputation and endorsement for web services. In: *Special Interest Group on Electronic Commerce Exchanges* 3 (2002), Nr. 1, S. 24–31

[MT04] MAUTHE, Andreas U. ; THOMAS, Peter: *Professional Content Management Systems: Handling Digital Media Assets*. John Wiley & Sons, 2004

[MW05] MOSCIBRODA, Thomas ; WATTENHOFER, Roger: Facility location: distributed approximation. In: *Tagungsband 24th Annual ACM Symposium on Principles of Distributed Computing*, ACM Press, 2005, S. 108–117

[Mül05] MÜLLER, Joachim: *Workflow-based Integration: Grundlagen, Technologien, Management*. Springer-Verlag, 2005

[NCL+03] NEAL, Stephen W. ; COLE, James B. ; LININGTON, Peter F. ; MILOSEVIC, Zoran ; GIBSON, Simon ; KULKARNI, Sachin: Identifying requirements for Business Contract Language: a Monitoring Perspective. In: *Tagungsband 7th International Conference on Enterprise Distributed Object Computing*, 2003, S. 50

[NERS08] NIEMANN, Michael ; ECKERT, Julian ; REPP, Nicolas ; STEINMETZ, Ralf: Towards a Generic Governance Model for Service-oriented Architectures. In: *Tagungsband 14th Americas Conference on Information Systems*, 2008

[NGG+09] NADALIN, Anthony ; GOODNER, Marc ; GUDGIN, Martin ; BARBIR, Abbie ; GRANQVIST, Hans ; LAWRENCE, Kelvin (Hrsg.) ; KALER, Chris (Hrsg.): *WS-SecurityPolicy 1.3*. OASIS Web Services Secure Exchange Technical Committee, 2009 `http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/ws-securitypolicy.html` – abgerufen am 05.04.2009.

[Nwa96] NWANA, Hyacinth S.: Software Agents: An Overview. In: *Knowledge Engineering Review* 11 (1996), November, Nr. 3, S. 1–40

[NZ01] Ng, T. S. E. ; Zhang, Hui: Towards global network positioning. In: *Tagungsband 1st ACM SIGCOMM Workshop on Internet Measurement*, 2001, S. 25–29

[Olb08] Olbrich, Alfred: *ITIL kompakt und verständlich: Effizientes IT Service Management*. Vieweg+Teubner, 2008

[ON98] O'Brien, Paul D. ; Nicol, Richard C.: FIPA – Towards a Standard for Software Agents. In: *BT Technology Journal* 16 (1998), Nr. 3, S. 51–59

[On04] On, Giwon: *Quality of Availability for Widely Distributed and Replicated Content Stores*, Technische Universität Darmstadt, Dissertation, Juni 2004

[OSS02] On, Giwon ; Schmitt, Jens ; Steinmetz, Ralf: On Availability QoS for Replicated Multimedia Service and Content. In: *Tagungsband Joint International Workshops on Interactive Distributed Multimedia Systems and Protocols for Multimedia Systems*, 2002, S. 313–326

[OT05] Overhage, Sven ; Thomas, Peter: WS-Specification: Ein Spezifikationsrahmen zur Beschreibung von Web-Services auf Basis des UDDI-Standards. In: Ferstl, O.K. (Hrsg.) ; Sinz, E.J. (Hrsg.) ; Eckert, S. (Hrsg.) ; Isselhorst., T. (Hrsg.): *Tagungsband Konferenz Wirtschaftsinformatik*, Physica Verlag, 2005, S. 1539–1558

[Ove02] Overhage, Sven: On Specifying Web Services Using UDDI Improvements. In: *Tagungsband Konferenz Net.ObjectDays*, 2002, S. 535–550

[OW02] Ottmann, Thomas ; Widmayer, Peter: *Algorithmen und Datenstrukturen*. Spektrum Akademischer Verlag, 2002

[Pap03] Papazoglou, Mike P.: Service-Oriented Computing: Concepts, Characteristics and Directions. In: *Tagungsband 4th International Conference on Web Information Systems Engineering*, 2003, S. 3–12

[PHA05] Pautasso, Cesare ; Heinis, Thomas ; Alonso, Gustavo: Autonomic Execution of Web Service Compositions. In: *Tagungsband IEEE International Conference on Web Services*, 2005, S. 435–442

[Pla06] Plaxton, C. G.: Approximation algorithms for hierarchical location problems. In: *Journal of Computer and System Sciences* 72 (2006), Nr. 3, S. 425–443

[Pro98] Programme, United Nations D.: *United Nations Standard Products and Services Code*. 1998

[PSL03] Patel, Chintan ; Supekar, Kaustubh ; Lee, Yugyung: A QoS Oriented Framework for Adaptive Management of Web Service Based Workflows. In: *Tagungsband 4th International Conference on Database and Expert Systems Applications*, 2003, S. 826–835

[QPV01] Qiu, Lili ; Padmanabhan, Venkata N. ; Voelker, Geoffrey M.: On the Placement of Web Server Replicas. In: *Tagungsband IEEE INFOCOM*, 2001, S. 1587–1596

[Ran03] Ran, Shuping: A model for web services discovery with QoS. In: *Special Interest Group on Electronic Commerce Exchanges* 4 (2003), Nr. 1, S. 1–10

[RBE+07] Repp, Nicolas ; Berbner, Rainer ; Eckert, Julian ; Heckmann, Oliver ; Schulte, Stefan ; Steinmetz:, Ralf: Der Einfluss von Transportschicht-Anomalien auf die Performanz von Web Services. In: *Tagungsband 5. Fachtagung Kommunikation in Verteilten Systemen*, 2007, S. 117–122

[RBHS07] Repp, Nicolas ; Berbner, Rainer ; Heckmann, Oliver ; Steinmetz, Ralf: A Cross-Layer Approach to Performance Monitoring of Web Services. In: *Emerging Web Services Technology*, 2007, S. 21–32

[Rep08] Repp, Nicolas: Monitoring of Services in Distributed Workflows. In: *Tagungsband 3rd International Conference on Software and Data Technologies*, 2008, S. 15–25

[Res00] Rescorla, Eric: *RFC 2818 – HTTP Over TLS*. IETF Network Working Group, 2000 `http://www.ietf.org/rfc/rfc2818.txt` – abgerufen am 05.04.2009.

[RES+08] Repp, Nicolas ; Eckert, Julian ; Schulte, Stefan ; Niemann, Michael ; Berbner, Rainer ; Steinmetz:, Ralf: Towards Automated Monitoring and Alignment of Service-based Workflows. In: *Tagungsband IEEE International Conference on Digital Ecosystems and Technologies*, 2008, S. 235–240

[RH06] Reussner, Ralf ; Hasselbring, Wilhelm ; Reussner, Ralf (Hrsg.) ; Hasselbring, Wilhelm (Hrsg.): *Handbuch der Software-Architektur*. dpunkt.verlag, 2006

[Ric92] Richter, Manfred: *Ein Rauschgenerator zur Gewinnung von quasi-idealen Zufallszahlen für die stochastische Simulation*. Shaker-Verlag, 1992

[RM08] Repp, Nicolas ; Martin, Wolfgang: *SOA Check 2008: Status Quo und Trends im Vergleich zum SOA Check 2007*. IT-Verlag, 2008

[RMNS08] Repp, Nicolas ; Miede, André ; Niemann, Michael ; Steinmetz, Ralf: WS-Re2Policy: A policy language for distributed SLA monitoring and enforcement. In: *Tagungsband 3rd International Conference on Systems and Networks Communications*, 2008, S. 256–261

[RMS08] Repp, Nicolas ; Mauthe, Andreas U. ; Steinmetz, Ralf: Chancen und Risiken bei der Einführung von IT-Governance-Frameworks. In: *IT-Governance* 2 (2008), März, Nr. 3, S. 9–14

[Rob05] Robinson, William N.: A requirements monitoring framework for enterprise systems. In: *Journal of Requirements Engineering* 11 (2005), Nr. 1, S. 17–41

[RR04] Rosenberg, Jothy ; Remy, David: *Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption*. Sams, 2004

[RSE+07b] Repp, Nicolas ; Schulte, Stefan ; Eckert, Julian ; Berbner, Rainer ; Steinmetz, Ralf: Service-Inventur: Aufnahme und Bewertung eines Services-Bestands. In: Steffens, Ulrike (Hrsg.) ; Addiks, Jan S. (Hrsg.) ; Streekmann, Niels (Hrsg.): *Tagungsband Workshop MDD, SOA und IT-Management*, 2007, S. 13–22

[13] Repp, Nicolas ; Schuller, Dieter ; Siebenhaar, Melanie ; Miede, André ; Niemann, Michael ; Steinmetz, Ralf: On distributed SLA monitoring and enforcement in service-oriented systems. In: *International Journal On Advances in Systems and Measurements* 2 (2009), Nr. 1 – erscheint Ende Mai 2009

[SBM05] Steinmetz, Ralf ; Berbner, Rainer ; Martinovic, Ivan: Web Services zur Unterstützung flexibler Geschäftsprozesse in der Finanzwirtschaft. In: Sokolovsky, Zbynek (Hrsg.) ; Löschenkohl, Sven (Hrsg.): *Industrialisierung der Finanzwirtschaft*, Gabler-Verlag, Dezember 2005, S. 641–654

[SBS+09] Scholz, Andreas ; Buckl, Christian ; Sommer, Stephan ; Kemper, Alfons ; Knoll, Alois ; Heuer, Jörg ; Schmitt, Anton: eSOA – SOA für eingebettete Netze. In: *Tagungsband Workshops der Wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen*, 2009, S. 1–9

[Sch96] Schulte, Roy W.: *SSA Research Note SPA-401-069, Service Oriented Architectures, Part 2*. Gartner Group, 1996

[Sch01] Schmitt, Jens B.: *Heterogeneous Network Quality of Service Systems*. Kluwer Academic Publishers, 2001

[SDM01] Sahai, Akhil ; Durante, Anna ; Machiraju, Vijay: Towards Automated SLA Management for Web Services / HP Laboratories Palo Alto. 2001 (HPL-2001-310) – Technical Report

[SDS05] Schmietendorf, Andreas ; Dumke, Reiner R. ; Stojanov, Stanimir: Performance aspects in Web Service-based Integration Solutions. In: *Tagungsband 21st UK Performance Engineering Workshop*, 2005, S. 137–152

[Ser05] Service Centric Systems Engineering Consortium: *State of the art on service delivery*. Service Centric Systems Engineering Consortium, 2005

[SERS08] Schulte, Stefan ; Eckert, Julian ; Repp, Nicolas ; Steinmetz, Ralf: An Approach to Evaluate and Enhance the Retrieval of Semantic Web Services. In: *Tagungsband 5th International Conference on Service Systems and Service Management*, 2008, S. 237–243

[SFBH+05] Schumacher, Markus ; Fernandez-Buglioni, Eduardo ; Hybertson, Duane ; Buschmann, Frank ; Sommerlad, Peter: *Security Patterns: Integrating Security and Systems Engineering*. Wiley & Sons, 2005

[SGKT05] Suh, Kyoungwon ; Guo, Yang ; Kurose, Jim ; Towsley, Don: Locating network monitors: complexity, heuristics, and coverage. In: *Tagungsband 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, 2005, S. 351–361

[SKR+08] Schmitt, Johannes ; Kropff, Matthias ; Reinhardt, Andreas ; Hollick, Matthias ; Schäfer, Christian ; Remetter, Frank ; Steinmetz, Ralf: An Extensible Framework for Context-aware Communication Management Using Heterogeneous Sensor Networks / Technische Universität Darmstadt. 2008 (TR-KOM-2008-08) – Technical Report

[SL05] Song, Hyung G. ; Lee, Kangsun: sPAC (Web Services Performance Analysis Center): Performance Analysis and Estimation Tool of Web Services. In: Aalst, Wil M. P. d. (Hrsg.) ; Benatallah, Boualem (Hrsg.) ; Casati, Fabio (Hrsg.) ; Curbera, Francisco (Hrsg.): *Tagungsband 3rd International Business Process Management Conference* Bd. 3649 / 2005, Springer-Verlag, September 2005 (Lecture Notes in Computer Science), S. 109–119

[SL08] Schubert, Petra ; Leimstoll, Uwe: How SMEs Strive to Achieve Competitive Advantage with IT-Supported Business Processes: An Empirical Study. In: *Tagungsband 21st International Bled eConference*, 2008, S. 114–127

[SM06] Spanoudakis, George ; Mahbub, Khaled: Non Intrusive Monitoring of Service Based Systems. In: *International Journal of Cooperative Information Systems* 15 (2006), Nr. 3, S. 325–358

[SN96] Schulte, Roy W. ; Natis, Yefim V.: *SSA Research Note SPA-401-068, Service Oriented Architectures, Part 1*. Gartner Group, 1996

[SN07] Steinmetz, Ralf ; Nahrstedt, Klara: *Multimedia Systems*. Springer-Verlag, 2007

[Som04] Sommerville, Ian: *Software Engineering*. 7. Addison-Wesley Longman, 2004

[SR01] SCHUMACHER, Markus ; ROEDIG, Utz: Security Engineering with Patterns. In: *Tagungsband 8th Conference on Pattern Languages of Programs*, 2001, S. 1–17

[SRS+07] SCHULTE, Stefan ; REPP, Nicolas ; SCHAARSCHMIDT, Ralf ; ECKERT, Julian ; BERBNER, Rainer ; STEINMETZ, Ralf ; BLANCKENBURG, Korbinian von: *Service-orientierte Architekturen: Status quo und Perspektive für die deutsche Bankenbranche*. Books on Demand, 2007

[SRSS09] SCHULTE, Stefan ; REPP, Nicolas ; SCHULLER, Dieter ; STEINMETZ, Ralf: From Web Service Policies to Automatic Deviation Handling: Supporting Semantic Description of Countermeasures to Policy Violations. In: *Tagungsband 3rd IEEE International Conference on Semantic Computing*, 2009 – im Begutachtungsprozess

[SS07] SCHILL, Alexander ; SPRINGER, Thomas: *Verteilte Systeme*. Springer-Verlag, 2007

[STA97] SHMOYS, David B. ; TARDOS Éva ; AARDAL, Karen: Approximation algorithms for facility location problems (extended abstract). In: *Tagungsband 29th Annual ACM Symposium on Theory of computing*, 1997, S. 265–274

[Sta99] STALLINGS, William: *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Addison-Wesley Longman, 1999

[Ste00] STEINMETZ, Ralf: *Multimedia-Technologie: Grundlagen, Komponenten und Systeme*. Springer-Verlag, 2000 (3)

[SW05] STEINMETZ, Ralf ; WEHRLE, Klaus: *Peer-to-Peer Systems and Applications*. Springer-Verlag, 2005

[Tan02] TANENBAUM, Andrew S.: *Computer Networks, Fourth Edition*. Prentice Hall, 2002

[TGNR03] TIAN, Min ; GRAMM, Andreas ; NAUMOWICZ, Thiemo ; RITTER, Hartmut: A concept for QoS integration in Web services. In: *Tagungsband 4th International Conference on Web Information Systems Engineering Workshops*, 2003, S. 149–155

[Tha01] THATTE, Satish: *XLANG Web Services for Business Process Design*. Microsoft Cooperation, 2001 `http://xml.coverpages.org/XLANG-C-200106.html` – abgerufen am 05.04.2009.

[TME+06] TOSIC, Vladimir ; MA, Wei ; ESFANDIARI, Babak ; PAGUREK, Bernard ; LUTFIYYA, Hanan: Extending Apache Axis for Monitoring of Web Service Offerings. In: *International Journal of Cases on Electronic Commerce* 2 (2006), Nr. 3, S. 53–75

[TPP02] TOSIC, Vladimir ; PATEL, Kruti ; PAGUREK, Bernard: WSOL – Web Service Offerings Language. In: *Tagungsband International Workshop on Web Services, E-Business, and the Semantic Web*, Springer-Verlag, 2002, S. 57–67

[TPP03] TOSIC, Vladimir ; PAGUREK, Bernard ; PATEL, Kruti: WSOL – A Language for the Formal Specification of Various Constraints and Classes of Service for Web Services. In: *Tagungsband International Conference On Web Services*, 2003, S. 375–381

[TR00] THEILMANN, Wolfgang ; ROTHERMEL, Kurt: Dynamic Distance Maps of the Internet. In: *Tagungsband 19th Conference on Computer Communications*, 2000, S. 275–284

[Tur09] TURCZYK, Lars A.: *Information Lifecycle Management – Eine Methode zur Wertzuweisung von Dateien*, Technische Universität Darmstadt, Dissertation, April 2009

[ULMS08] Unger, Tobias ; Leymann, Frank ; Mauchart, Stephanie ; Scheibler, Thorsten: Aggregation of Service Level Agreements in the Context of Business Processes. In: *Tagungsband 12th International IEEE Enterprise Distributed Object Computing Conference*, 2008, S. 43–52

[Vaz01] Vazirani, Vijay V.: *Approximation Algorithms*. Springer-Verlag, 2001

[WC95] Widom, Jennifer ; Ceri, Stefano: *Active Database Systems*. Morgan-Kaufmann, 1995

[Wes07] Weske, Mathias: *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag, 2007

[Win03] Winer, Dave: *XML-RPC Specification*. UserLand Software, 2003 `http://www.xmlrpc.com/spec/` – abgerufen am 05.04.2009.

[WM08] White, Stephen A. ; Miers, Derek: *BPMN Modeling and Reference Guide: Understanding and Using BPMN*. Future Strategies, 2008

[Woo04] Woods, Dan: *Enterprise Services Architecture*. Galileo Press, 2004

[WS06] Wilson, Kirk (Hrsg.) ; Sedukhin, Igor (Hrsg.): *Web Services Distributed Management: Management of Web Services (WSDM-MOWS) 1.1*. OASIS Web Services Distributed Management TC, 2006 `http://docs.oasis-open.org/wsdm/wsdm-mows-1.1-spec-os-01.pdf` – abgerufen am 05.04.2009.

[WZZ06] Wu, Ling-Yun ; Zhang, Xiang-Sun ; Zhang, Ju-Liang: Capacitated facility location problem with general setup cost. In: *Computers & Operations Research* 33 (2006), S. 1226–1241

[XN99] Xiao, Xipeng ; Ni, Lionel M.: Internet QoS: A Big Picture. In: *IEEE Network* 13 (1999), März/April, Nr. 2, S. 8–18

[XX08] Xu, Guang ; Xu, Jinhui: An improved approximation algorithm for uncapacitated facility location problem with penalties. In: *Journal of Combinatorial Optimization* (2008)

[XZ07] Xu, Dachuan ; Zhang, Shuzhong: An Improved Approximation Algorithm for the Uncapacitated Facility Location Problem with Service Installation Costs / Chinese University of Hong Kong. 2007 (SEEM2007-03) – Technical Report

[ZHG99] Zapf, Michael ; Herrmann, Klaus ; Geihs, Kurt: Decentralized SNMP Management with Mobile Agents. In: *Tagungsband 6th IFIP/IEEE International Symposium on Integrated Network Management*, 1999, S. 623–635

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

ACID	Atomicity, Consistency, Isolation, Durability
ACL	Agent Communication Language
AJAX	Asynchronous JavaScript and XML
AMAS	Automated Monitoring and Alignment of Services
AOP	Aspekt-orientierte Programmierung
API	Application Programming Interface
BCL	Business Contract Language
BDI	Belief, Desire, Intention
BGP	Border Gateway Protocol
BPEL	Business Process Execution Language
BPEL4People	WSBPEL Extension for People
BPEL4WS	Business Process Execution Language for Web Services
BPMN	Business Process Modeling Notation
BPO	Business Process Outsourcing
BRITE	Boston university Representative Internet Topology gEnerator
CDN	Content Distribution Networks
CIM	Common Information Model
CORBA	Common Object Request Broker Architecture
DiffServ	Differentiated Services Architecture
EAI	Enterprise Application Integration
ECA	Event, Condition, Action
ESB	Enterprise Service Bus
FIPA	Foundation for Intelligent, Physical Agents
FLP	Facility Location Problem
FMC	Fundamental Modeling Concepts
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IDL	Interface Description Language
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IIOB	Internet Inter-ORB Protocol
IntServ	Integrated Services Architecture
IoS	Internet of Services
IP	Internet Protocol
IT	Informationstechnologie
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
JADE	Java Agent Development Framework
JMS	Java Message Service
MAA	Monitoring & Alignment Agent
MAU	Monitoring & Alignment Unit
MULP	Monitoring Unit Location Problem

O2A	Object to Agent Communication
OMG	Object Management Group
P2P	Peer-to-Peer
QoE	Quality of Experience
QoS	Quality of Service
REST	Representational State Transfer
RFC	Request for Comments
RMI	Remote Method Invocation
RMON	Remote Monitoring
RPC	Remote Procedure Call
SA	Simulated Annealing
SAAJ	SOAP with Attachments for Java
SAIS	Simulated Annealing-based Improved Solution Heuristic
SAML	Security Assertion Markup Language
SLA	Service Level Agreements
SLO	Service Level Objective
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOA	Service-oriented Architecture
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TPS	Tree-based Primary Solution Heuristic
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WfMS	Workflow-Management-System
WLP	Warehouse Location Problem
WS	Webservice
WSBPEL	Web Service Business Process Execution Language
WS-CDL	Web Service Choreography Description Language
WSDL	Web Service Description Language
WSFL	Web Services Flow Language
WS-HumanTask	Web Services Human Task
WSLA	Web Service Level Agreements
WSQoSX	Web Service Quality of Service Architectural Extensions
WS-Re2Policy	Web Service Requirements and Reaction Policy Language
XML	Extensible Markup Language
XPath	XML Path Language

A

ANHANG

A.1 GRUNDLAGEN DER WEBSERVICE-TECHNOLOGIE

A.1.1 *Definition*

Als Grundlage für die Diskussion des Webservice-Begriffs wird an dieser Stelle eine der am weitest verbreiteten Definition angeführt, die von einer Arbeitsgruppe des W3C entwickelt wurde [ABFG04]:

„A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols."

Dem Verständnis der W3C nach sind Webservices Softwaresysteme, die unter Verwendung einer eindeutigen Adresse über das Internet aufgerufen werden können. Zur Beschreibung der Funktionalität eines solchen Softwaresystems, zum Auffinden des Systems im Internet sowie zur Übertragung der Daten kommen XML-basierte Sprachen und Datenaustauschformate zum Einsatz.

Betrachtet man allerdings die Verwendung des Begriffs in der Literatur sowie im praktischen Gebrauch, so lässt sich feststellen, dass das Verständnis des Begriffs „Webservice" sehr ambivalent ist. Weit verbreitet ist die Ansicht, dass Webservices mit einer Reihe von Standards zur Unterstützung der Interoperabilität heterogener Systeme gleichzusetzen sind [Jos08], [HB04]. Zu diesen Standards, welche häufig mit *WS** bezeichnet werden, zählt die Web Service Description Language (WSDL), eine XML-basierte Sprache zur Beschreibung von Diensten, ebenso wie der UDDI-Standard (Universal Description, Discovery and Integration) zum Auffinden von Diensten sowie das Transportprotokoll SOAP (siehe Abbildung 29).

Ein WSDL-Dokument beschreibt für jeden Dienst dessen Funktionalität, den Ort bzw. die Adresse, an welcher der Dienst zu finden ist sowie die für seinen Aufruf notwendigen Parameter (die Schnittstelle des Dienstes). Das Auffinden eines Dienstes, der durch WSDL beschrieben ist, wird mit Hilfe von UDDI realisiert. UDDI bietet hierfür einen Verzeichnisdienst zur Verwaltung von Metainformationen über Dienste sowie eine Sammlung von standardisierten Schnittstellen, über welche sich Webservices suchen und einbinden lassen. Zum Austausch von Daten mit einem Webservice kommt SOAP[29] zum Einsatz. SOAP spezifiziert zu diesem Zweck ein XML-basiertes Protokoll für den Fernaufruf von Diensten über Web-Infrastrukturen sowie die für die Übertragung benötigten Codierungen von plattformabhängigen Daten in XML.

Für das Verständnis der Arbeit ist es wichtig, dass Webservices nicht ausschließlich als Implementierungen der eben aufgeführten Standards verstanden werden,

29 Ursprünglich wurde der Begriff „SOAP" als Abkürzung für das „Simple Object Access Protocol" genutzt – mittlerweile wird SOAP aber als eigenständiger Begriff gesehen und nicht mehr als Abkürzung verwendet.

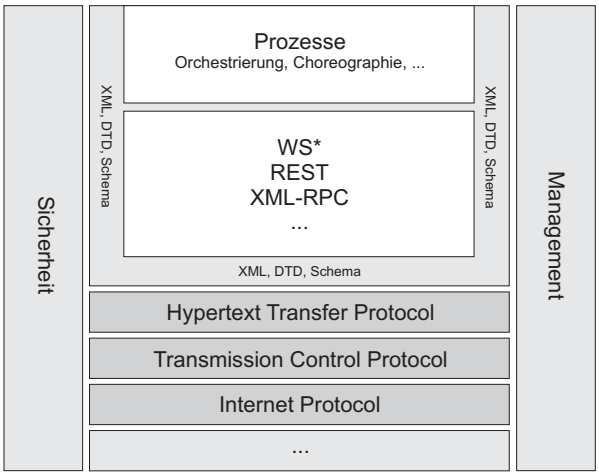


Abbildung 29: Webservice-Standards und Technologien [BHM+04], [RBHS07]

sondern vielmehr im Sinne der weiter gefassten Definition des W3C. Es wird explizit keine Einschränkung auf einzelne Standards vorgenommen. Im Sinne dieser Arbeit stellen Webservices Dienste innerhalb eines verteilten Systems auf Basis der Nutzung von Webtechnologien zur Verfügung. Hierbei wird unter Webtechnologien primär der Einsatz von HTTP als Transportmechanismus verstanden, über den ein Dienstaufruf sowie dessen Antwort transportiert wird. Zur Bereitstellung von Diensten können auch andere Transportmechanismen, wie z. B. das Simple Mail Transfer Protocol (SMTP) oder der Java Message Service (JMS), zum Einsatz kommen [Jos08], [ACKM04]. Diese spielen aber in der Praxis nur eine untergeordnete Rolle. Webservices setzen auf den bestehenden Web-Infrastrukturen auf, verwenden deren Routing-Mechanismen und nutzen beispielsweise deren Sicherheits- und Transaktionskonzepte. Auch ermöglichen Webservices auf XML-Basis, dass Firewall-Systeme die bei unternehmensübergreifenden Dienstaufrufen ausgetauschten Daten kontrollieren können.

Auf Grundlage dieses Verständnisses lassen sich weitere Technologien als Webservices verstehen, welche innerhalb bestimmter Anwendungsfelder eine Existenzberechtigung besitzen. Ebenfalls HTTP nutzen so genannte REST-Webservices (Representational State Transfer – auch als *RESTful Services* bezeichnet), die durch Verwendung des HTTP-Befehlssatzes auf Webressourcen navigieren und somit Daten per Funktionsaufruf zur Verfügung stellen können. Auf REST basierende Webservices finden im Bereich des Web 2.0 als Komplementärtechnologie zu AJAX (Asynchronous JavaScript and XML) verstärkt Einsatz. Gleichermaßen eine Nische unter den Webservices besetzt XML-RPC (Extensible Markup Language Remote Procedure Call). XML-RPC ist vom Aufbau und der Funktionsweise mit SOAP vergleichbar, verwendet aber einen wesentlich geringeren Sprachumfang, was seine Nutzung effizienter gestaltet. Weder *RESTful Services* noch XML-RPC spielen innerhalb der Arbeit eine Rolle. Für eine detaillierte Darstellung von REST wird an dieser Stelle auf [FT00], [Fie00] und [FT02] verwiesen. Den Einsatz von XML-RPC als Webservice-Standard ist Gegenstand von [Win03] und [Cer02].

Von der Zielsetzung her lassen sich Webservices auch mit anderen verteilten Softwarearchitekturen für Komponenten wie CORBA (Common Object Request Broker

Architecture) der Object Management Group (OMG) oder Jini von Sun vergleichen. Webservices unterscheiden sich primär von diesen durch die Art des Einsatzes von Webtechnologien und offenen Standards zur Ermöglichung von Interoperabilität. Zwar unterstützen sowohl CORBA mit dem Internet Inter-ORB Protocol (IIOB) als auch Jini die Nutzung grundlegender Webtechnologien. Diese basieren allerdings nicht auf offenen Standards zum Austausch sowie der Beschreibung von Diensten.

A.1.2 *Webservice-Standards und ergänzende Technologien*

SOAP

SOAP ist eine XML-basierte Sprache, die eine plattformunabhängige Nutzung von Webservices ermöglicht. Zu diesem Zweck definiert SOAP ein Protokoll für den Austausch SOAP-konformer Nachrichten zwischen einem Dienstanbieter und einem Dienstnutzer. SOAP ist vom W3C als Empfehlung, d. h. einer Vorstufe zu einem Standard, publiziert und wird in drei Teilen spezifiziert [ML07], [GHM+07a], [GHM+07b]. SOAP liegt aktuell in Version 1.2 vor.

Kerngegenstand, der von SOAP spezifiziert wird, ist die SOAP-Nachricht. Eine SOAP-Nachricht besteht aus einem *SOAP Envelope*, d. h. einem Umschlag, der die Nachricht begrenzt, einem *SOAP Header* sowie dem *SOAP Body*, welcher den eigentlichen Inhalt der Nachricht beinhaltet [MES+08], [ML07].

Die Verwendung des *SOAP Header* ist nicht verpflichtend. Innerhalb des Headers können beliebige Daten übertragen werden. Aus diesem Grund wird der Header auch nicht weiter durch das W3C spezifiziert. Häufig findet der *SOAP Header* allerdings Anwendung beim Transport sicherheitsrelevanter Daten, wie z. B. Informationen über die Verschlüsselungsverfahren, die auf den *SOAP Body* angewendet werden.

Im Gegensatz zum *SOAP Header* ist die Verwendung des *SOAP Body* verpflichtend, da in ihm die Nutzlast der SOAP-Nachricht transportiert wird. Zur Übertragung wird eine Serialisierung bzw. Deserialisierung von Daten bei Dienstanbieter und -nutzer in ein XML-Format benötigt [Erl05]. Das zum Einsatz kommende Format zur Codierung wird im *SOAP Envelope* definiert und gilt für die gesamte Nachricht. Die Inhalte des *SOAP Body* hängen vom gewählten Typ der SOAP-Nutzung ab. SOAP unterstützt zwei Arten der Verarbeitung von Nachrichten (Interaktionsmuster), welche als *document style* und *RPC style* bezeichnet werden [ACKM04]. Ein SOAP-basierter Webservice, der im Verarbeitungsmodus *document style* benutzt wird, überträgt die Nutzlast im *SOAP Body* in einem oder mehreren frei wählbaren XML-Formaten. Die Interpretation der beim Dienstaufrufe und den Rückantworten übertragenen XML-Dokumente ist Aufgabe der Applikationen, die den Dienst anbieten bzw. nutzen. Im Gegensatz hierzu verwenden Dienste, die dem *RPC style* folgen eine eindeutig definierte Codierung in ein XML-Datenformat, welches in der SOAP-Empfehlung spezifiziert ist [GHM+07b], [MES+08]. Es wird an Stelle eines Austauschs von vollständigen Dokumenten ein Fernaufruf einer Methode eines Dienstes mit entsprechenden Aufrufparametern durchgeführt und nachfolgend auf eine spezifizierte Antwort des Dienstes gewartet.

Ebenfalls sieht die SOAP-Spezifikation Mechanismen für die Behandlung von Fehlern und anderen Ausnahmezuständen vor, die während der Dienstnutzung auftreten können. Die so genannten *SOAP Faults* erlauben die Signalisierung von Aus-

nahmezuständen innerhalb einer SOAP-Nachricht zur Information des Empfängers einer Nachricht [GHM+07b]. *SOAP Faults* kommen beispielsweise zum Einsatz, um eine Abstimmung zwischen Sender und Empfänger einer Nachricht über den notwendigen Sprachumfang zu ermöglichen. Ein Sender kann durch das Setzen des Attributs *mustUnderstand* im Header einer SOAP-Nachricht signalisieren, dass ein bestimmtes Feld zur Verarbeitung der Nachricht notwendig ist. Sollte der Empfänger einer solchen Nachricht das Feld nicht verarbeiten können, so gibt er einen entsprechenden *SOAP Fault* zurück.

Zur Übertragung der SOAP-Nachrichten über das Internet wird am häufigsten HTTP als Protokoll verwendet, auch wenn HTTP nicht verpflichtend von der SOAP-Spezifikation vorgegeben wird [Jos08]. Es existieren in der Spezifikation weitere Bindungen an andere Transportmechanismen. Beispielsweise kommt der Nutzung von SOAP über HTTPS sowie JMS eine wachsende Bedeutung zu [BBF+06]. Um die Kommunikation mit einem Webservice auf Basis von SOAP durchführen zu können, muss weiterhin der Ort bekannt sein, an dem sich der Webservice befindet. Innerhalb einer SOAP-Nachricht wird deshalb die URL (Uniform Resource Locator) des Dienstes als so genannter *Service Endpoint* spezifiziert. Der *Service Endpoint* identifiziert die den Dienst ausführende Instanz in eindeutiger Form.

Web Service Description Language (WSDL)

WSDL ist eine XML-basierte Sprache zur Beschreibung von Schnittstellen eines Webservices. Die Sprache WSDL lässt sich aus funktionaler Sicht mit der Interface Description Language (IDL) von CORBA vergleichen [SS07], [Bir05], basiert allerdings im Vergleich zu dieser ausschließlich auf XML.

WSDL wird vom W3C spezifiziert und liegt aktuell in ihrer zweiten Version vor [BL07], [CMRW07], [CMRW07]. WSDL ermöglicht die Beschreibung von Schnittstellen unabhängig von der konkreten Implementierungstechnologie des Dienstes. Innerhalb eines zu einem Dienst gehörenden WSDL-Dokuments werden die Operationen bzw. Methoden des Dienstes sowie die zum Aufruf notwendigen Parameter spezifiziert. Darüber hinaus wird festgelegt, durch welche Protokolle der Dienst genutzt werden kann.

WSDL-Dokumente folgen dem WSDL 2.0 Komponentenmodell [BL07], dessen Struktur sich im Aufbau eines WSDL-Dokuments widerspiegelt. Ein WSDL-konformes Dokument beinhaltet innerhalb des *Description* Elements die Beschreibung WSDL-spezifischer Definitionen als auch Typdefinitionen (*Types*), welche unabhängig von WSDL sein können. Typdefinitionen stellen einen verpflichtenden Teil innerhalb des WSDL-Dokuments dar, welcher die vom Webservice genutzten Datentypen spezifiziert. Zu den Typdefinitionen zählen neben Datentypen auch die Nachrichtenformate, welche von Diensten verwendet werden. Entsprechende Typdefinitionen werden beispielsweise auf Basis von XML Schema spezifiziert.

Die WSDL-spezifischen Komponenten können weiterhin in Komponenten mit abstrakt und konkret spezifizierten Inhalten unterschieden werden. Hintergrund einer solchen Unterscheidung ist die Wiederverwendbarkeit von abstrakt spezifizierten Inhalten in unterschiedlichen Kontexten. Zu den Komponenten, die abstrakt spezifizierte Inhalte enthalten, zählen die zuvor beschriebenen *Types* sowie die *Interfaces*. Konkret spezifizierte Inhalte enthalten die Komponenten *Bindings*, *Endpoints* und

Services. Die einzelnen WSDL-spezifischen Komponenten werden nachfolgend im Detail erläutert [BL07]:

- *Interfaces* stellen abstrakte Beschreibungen von Operationen dar, die ein Dienst anbietet. Operationen werden aus der Folge von auszutauschenden Eingangs- und Ausgangsnachrichten gebildet und folgen vorgegebenen Kommunikationsprimitiven (*Message Exchange Patterns*), die nachfolgend gesondert beschrieben werden.
- *Bindings* definieren die Zuordnung der zu verwendenden Übertragungsprotokolle sowie Datenformate zu Nachrichten oder Operationen für ein gegebenes *Interface*.
- *Endpoints* spezifizieren jeweils einen Einstiegspunkt in einen Dienst, bestehend aus der Verknüpfung eines *Bindings* mit einem URI, der den Webservice eindeutig identifiziert.
- *Services* stellen Gruppierungen zusammengehörender *Endpoints* dar und bilden somit einen Webservice.

Das WSDL 2.0 Komponentenmodell spezifiziert darüber hinaus für die *Interfaces* und *Bindings* zusätzliche Operationen und Ausnahmezustände, auf die aber an dieser Stelle nicht weiter eingegangen werden soll.

Wie zuvor bereits beschrieben, verwenden *Interfaces* bzw. die durch sie definierten Operationen verschiedene Kommunikationsprimitive, welche in WSDL 2.0 als *Message Exchange Patterns* bezeichnet werden. Ein solches Pattern beschreibt das Zusammenspiel der Parteien innerhalb einer Interaktion sowie die dabei involvierten abstrakten Nachrichten.

Ebenfalls von hoher Relevanz sind die von WSDL unterstützten *Bindings*, d. h. Bindungen eines Dienstes, einer Operation oder einer Nachricht an existierende Protokolle und Datenformate. WSDL besitzt einen erweiterbaren Mechanismus für entsprechende Bindungen, so dass die Sprache jederzeit um neue Bindungskonstrukte erweitert werden kann. Die Spezifikation in Version 2.0 sieht bereits verschiedene Bindungen für Transportprotokolle vor, nämlich Bindungen für SOAP sowie HTTP. Im Vergleich zu WSDL 1.1 wird die vollständige HTTP-Spezifikation ebenso wie HTTPS (HTTP Secure – vgl. [Res00]) unterstützt, was eine Beschreibung von REST-Webservices durch WSDL möglich macht.

Abschließend gilt es anzumerken, dass WSDL nur die reine Syntax eines Dienstes beschreibt. Die semantische Beschreibung eines Webservice, seiner Aufrufparameter sowie anderer funktionaler und nicht-funktionaler Eigenschaften, wie beispielsweise den Dienstgütemerkmalen eines Dienstes, ist Gegenstand einer Reihe von Spracherweiterungen. WSDL-S (WSDL-Semantic – vgl. [AFM+05]) und OWL-S (Ontology Web Language-Services – vgl. [BHL+04]) definieren beispielsweise zur Beschreibung von Webservices notwendige Ontologien.

Universal Description, Discovery and Integration (UDDI)

Mit Hilfe der UDDI-Spezifikation ist es möglich, Webservices zu dokumentieren und zu verwalten (Description), zu finden (Discovery) und in eigene Anwendungen zu integrieren (Integration). Die UDDI-Spezifikation wurde in einer ersten Version

im Jahr 2000 durch ein Industriekonsortium um Microsoft, IBM und Ariba spezifiziert. Die aktuell gültige dritte Version der Spezifikation stammt aus dem Jahr 2005 und ist durch OASIS als Standard verabschiedet worden. In den letzten Jahren haben verschiedene Betreiber von UDDI-Verzeichnisdiensten ihre Registries abgeschaltet. De-facto wird die UDDI-Spezifikation in Ermangelung einer Alternative immer noch als Austauschformat für Dienstbeschreibungen zwischen Dienstverwaltungssystemen unterschiedlicher Hersteller eingesetzt.

Die UDDI-Spezifikation beschreibt einen Verzeichnisdienst, mit dessen Hilfe Webservices verwaltet werden können. Sie beschreibt ein Datenmodell, auf Grundlage dessen Webservices beschrieben werden können sowie einen Satz an Schnittstellen, unter deren Verwendung Webservices beim Verzeichnisdienst registriert, existierende Einträge gewartet und Dienste gefunden werden können. Ein UDDI-konformer Verzeichnisdienst beschreibt alle für eine Verwendung eines Webservices notwendigen Informationen. Hierzu zählen Angaben über den Ort eines Webservices sowie dessen Aufrufparameter ebenso wie Angaben über das den Dienst anbietende Unternehmen oder die Branche und die Prozesse, in welchen ein Dienst eingesetzt werden kann. Die Beschreibung der einzelnen Aspekte wird in so genannten „Pages“ abgebildet [CHRR04]:

- *White Pages* beinhalten beschreibende Informationen über das Unternehmen, welches den Webservice anbietet. Bestandteil der *White Pages* sind der Name des Unternehmens, Adress- und Kontaktinformationen. Darüber hinaus können weitere das Unternehmen identifizierende Informationen hinterlegt werden, wie z. B. eine weltweit eindeutige Unternehmenskennzeichnung auf Basis des Data Universal Numbering Systems oder die Umsatzsteuer-Identifikationsnummer des Unternehmens.

- *Yellow Pages* beschreiben die Branche, in der das anbietende Unternehmen tätig ist, nicht aber die dem Dienst zugeordnete Branche. Zum Einsatz kommen hierzu Taxonomien, wie z. B. der UN-Standard zur Klassifikation von Produkten und Dienstleistungen (United Nations Standard Products and Services Code – vgl. [Pro98]).

- *Green Pages* beschreiben den Webservice selbst. In den *Green Pages* sind Referenzen auf die Spezifikation eines Webservices hinterlegt sowie Referenzen auf die zu dessen Aufruf benötigten Informationen, wie beispielsweise Informationen über zu verwendende Protokolle und notwendige Aufrufparameter.

Die „Pages“ werden innerhalb eines UDDI-konformen Verzeichnisdienstes auf Basis eines XML-basierten Datenmodells dokumentiert (vgl. [CHRR04]).

Darüber hinaus existieren zahlreiche Erweiterungsvorschläge für UDDI, wie z. B die Erweiterung um semantische Eigenschaften (vgl. [Ove02] und [OT05]).

Business Process Execution Language for Web Services (WSBPEL)

Die Web Services Business Process Execution Language (WSBPEL oder kurz BPEL) ist eine XML-basierte Beschreibungssprache für Geschäftsprozesse, welche als Komposition von Webservices realisiert werden können [JE07], [ACKM04], [Erl05]. BPEL basiert auf der Web Services Flow Language (WSFL – vgl. [Ley01]) von IBM sowie

der Xlang von Microsoft [Tha01], welche beide zur Beschreibung von Webservice-basierten Geschäftsprozessen in den Produkten der jeweiligen Hersteller verwendet wurden. Die ersten Versionen der BPEL-Spezifikation wurden mit dem Akronym BPEL4WS (BPEL for Web Services) bezeichnet. Seit Version 2.0 der mittlerweile durch OASIS standardisierten Spezifikation wird die Bezeichnung WSBPEL offiziell verwendet.

BPEL wird hauptsächlich zur Komposition, genauer gesagt der Orchestrierung, von Webservices verwendet. Die Orchestrierung von Webservices beschreibt einen Prozess, der durch eine zentrale Instanz ausgeführt und kontrolliert wird [Jos08]. Im Gegensatz hierzu wird die zweite Form der Komposition, die so genannte Choreographie, zur dezentralen Interaktion von Webservices innerhalb eines verteilten Systems weit aus weniger häufig verwendet. Auch diese lässt sich in BPEL in Form abstrakter Prozesse abbilden oder aber durch Einsatz ergänzender Spezifikationen, wie beispielsweise der Web Service Choreography Description Language (WS-CDL – vgl. [KBR+05]), realisieren. Im Rahmen der Arbeit kommt BPEL allerdings ausschließlich zur Orchestrierung von Webservices zum Einsatz.

BPEL beschreibt das Zusammenspiel einzelner Aktivitäten zur Realisierung eines Geschäftsprozesses. Hierbei werden sowohl der Kontroll- als auch der Datenfluss innerhalb eines Prozesses unterstützt. Aktivitäten innerhalb eines Prozesses können in BPEL selbst beschrieben oder aber in Form von extern zur Verfügung stehenden Webservices eingebunden werden. Es ist somit möglich, mit BPEL Prozesse zu beschreiben und diese auszuführen, ohne dass zwangsläufig Webservices involviert sind. In der Praxis wird BPEL aber fast ausschließlich im Zusammenhang mit der Orchestrierung von Webservices verwendet. In BPEL beschriebene und durch eine Laufzeitumgebung, die so genannte BPEL-Engine, ausgeführte Geschäftsprozesse stellen ihre Funktionalität wiederum als Webservice zur Verfügung.

Kernelemente einer auf BPEL-basierenden Prozessbeschreibung sind, neben den aus klassischen Programmiersprachen stammenden Kontrollstrukturen, Datentypen und Variablen (bzw. Nachrichten), eine Reihe von Elementen, die eine enge Verknüpfung zu der zu Grunde liegenden Webservice-Technologie aufweisen. Im Besonderen macht BPEL von WSDL und dessen Komponentenmodell Gebrauch.

Der Aufbau eines in BPEL-spezifizierten Prozesses soll nachfolgend kurz erläutert werden. Die Anbieter eines Webservices werden in der BPEL-Spezifikation, ebenso wie der Prozessausführende selbst, als *Partner* definiert. Webservices selbst werden durch das Element *partnerLink* in einen Prozess eingebunden. Jeder *Partner* nimmt dabei innerhalb einer Interaktion eine bestimmte Rolle ein, die gesondert als Bestandteil eines *partnerLink* zu definieren ist. Durch die Spezifikation der Rollen ist die Unterscheidung der Interaktionsmuster in synchrone und asynchrone Dienstaufrufe möglich. Der Dienst wird anhand des dem *partnerLink* zugeordneten *partnerLinkType* referenziert, welcher auf ein den Webservice beschreibendes WSDL-Dokument verweist. Ein *partnerLinkType* beinhaltet verschiedene *portTypes*, die direkt auf die *Interface* Elemente innerhalb der WSDL-Beschreibung eines Dienstes abgebildet werden können. Die nachfolgend beschriebenen Aktivitäten können die *portTypes* nutzen, in dem sie die zugehörigen Operationen mit den entsprechenden in der WSDL beschriebenen Variablen als Parameter aufrufen.

Nach Einbindung der zu verwendenden Dienste als Aktivitäten können die unterschiedlichen Aktivitäten strukturiert ausgeführt werden, um die gewünschte Prozessfunktionalität abzubilden. Nachfolgend sollen ausgewählte von BPEL unterstütz-

te Aktivitäten kurz dargestellt werden – für eine vollständige Darstellung wird an dieser Stelle auf [JE07] verwiesen. Die Aktivitäten können in Basisaktivitäten und strukturierte Aktivitäten unterschieden werden [MES+08]. Zu den Basisaktivitäten zählen:

- *invoke* ruft einen Webservice auf
- *receive* ermöglicht den Aufruf des Prozesses als Webservice
- *reply* gibt die Ergebnisse des Prozesse zurück
- *throw* erzeugt eine Fehlermeldung

Strukturierte Aktivitäten steuern den eigentlichen Ablauf eines Prozesses. Zu ihnen gehören neben einfachen Kontrollstrukturen, wie z. B. *switch* und *while*, beispielsweise die nachfolgenden Aktivitäten:

- *sequence* führt Aktivitäten in Folge ihres Auftretens aus
- *flow* ermöglicht die parallele Ausführung von Aktivitäten

Darüber hinaus unterstützt der BPEL-Standard verschiedene Mechanismen zur Fehlerbehandlung innerhalb eines Geschäftsprozesses sowie Kompensationsmechanismen, welche das „Rücksetzen“ eines Prozesses bzw. dessen Zustand im Falle des Scheiterns einer Menge von Operationen erlauben.

Abschließend soll auf die zahlreichen Erweiterungen von BPEL hingewiesen werden. Die hohe Akzeptanz des Standards sowie die umfangreiche Unterstützung durch Plattformen und Werkzeuge hat eine Vielzahl von Erweiterungen zu BPEL und BPEL-unterstützenden Systemen motiviert, wie beispielsweise eine Ergänzung um Fähigkeiten zur Messung der Prozessperformanz [McG03] oder die Erweiterung um Fähigkeiten zur automatisierten Korrektur fehlerhafter Prozesse [MMP06], [PHA05]. Die Einbindung menschlicher Akteure in BPEL verfolgen die Initiativen BPEL4People (WSBPEL Extension for People – vgl. [AAD+07a]) und WS-HumanTask [AAD+07b].

A.2 GRUNDLAGEN MOBILER SOFTWAREAGENTEN

A.2.1 *Definition*

Für den Softwareagentenbegriff gibt es keine einheitliche Definition. Die nachfolgende Definition orientiert sich am Begriffsverständnis von [Nwa96], [BZW98] und [BCG07]. Agenten sind in diesem Zusammenhang Softwarekomponenten, welche die nachfolgenden Eigenschaften besitzen:

- *Autonomie*: Agenten können selbstständig auf Grund der ihnen gegebenen Fähigkeiten Entscheidungen treffen, ohne dass hierfür eine Intervention von Extern notwendig ist.
- *Persistenz*: Der Agentencode muss nicht von Extern aufgerufen werden. Vielmehr wird der Code permanent ausgeführt und bei Bedarf aktiv.

- *Reaktivität*: Agenten nehmen selbstständig ihr Umfeld wahr und reagieren auf Änderungen in diesem Umfeld.
- *Proaktivität*: Agenten nehmen aus Eigeninitiative Handlungen vor.
- *Kollaboration*: Agenten können mit anderen Agenten in Interaktion treten, um mit diesen gemeinsam Aufgaben zu lösen. Sie können sich untereinander koordinieren.

Als Analogie für einen Softwareagenten wird das menschliche Verhalten herangezogen. Agenten verhalten sich vergleichbar zu menschlichen Akteuren, die selbstständig für einen Auftraggeber Dienste erbringen [BCG07].

Je nach Definition sind verschiedene Ausprägungen von Agenten vorstellbar. So können Agenten durch Einsatz von Verfahren der künstlichen Intelligenz, wie z. B. Lernverfahren, zu intelligenten Agenten entwickelt werden. Agenten, die sich innerhalb eines verteilten Systems durch die Replikation von Code bewegen können, werden mobile Agenten genannt. Ebenfalls unterschieden werden Agenten anhand der Verhaltensmuster, welchen sie folgen. Rein reaktive Agenten entscheiden nicht, sondern agieren nur anhand vorgegebener Handlungsmuster und Regeln auf externe Einflüsse. Kognitive Agenten hingegen sind wesentlich komplexer, da sie auf einem Modell ihrer Umwelt operieren, auf dessen Basis sie planen und handeln können. Ebenfalls sind sie in der Lage, die zu Grunde liegende Wissensbasis zu erweitern bzw. zu verändern, um sich neuen Umwelteinflüssen anpassen zu können. In diese Kategorie fallen auch die weit verbreiteten BDI-Agenten (Belief, Desire, Intention), welche auf Grundlage einer Wissensbasis eigenständig Ziele verfolgen und dabei selbstständig zwischen Wegen zur Zielerreichung entscheiden können.

Softwareagenten stellen neben dem eben beschriebenen technologischen Konzept auch ein Entwurfsparadigma für verteilte Systeme dar, welches als agentenorientierte Softwareentwicklung bezeichnet wird.

A.2.2 *Spezifikation und Frameworkunterstützung*

In diesem Abschnitt wird die *Foundation for Intelligent, Physical Agents* Spezifikation vorgestellt, die den de-facto Standard für die Entwicklung agentenbasierter Systeme darstellt.

Die Arbeit an der FIPA-Spezifikation wurde im Jahre 1996 von einer Gruppe der IEEE Computer Society begonnen und im Jahre 2002 abgeschlossen. Sie stellt einen Standardisierungsversuch dar, um die im akademischen Umfeld verbreitete Agententechnologie für die industrielle Anwendung zu öffnen [BCG07], [ON98]. Die FIPA-Spezifikation vereinheitlicht eine Vielzahl von Bereichen, wobei die wichtigsten Aspekte die Kommunikation zwischen Agenten, das Management von Agenten sowie die zugehörige Architektur darstellen.

Zur Definition der Kommunikation wird von FIPA eine eigenständige *Agent Communication Language* (ACL) definiert. Weiterhin definiert FIPA verschiedene Sublayer auf der Anwendungsschicht des TCP/IP-Protokollstapels, welche sich beispielsweise mit der Codierung von Nachrichten sowie der semantischen Anreicherung der Nachrichteninhalte beschäftigen.

Das Management von Agenten wird innerhalb von FIPA durch verschiedene durch ein Framework zu implementierende Bestandteile eines Agentensystems definiert.

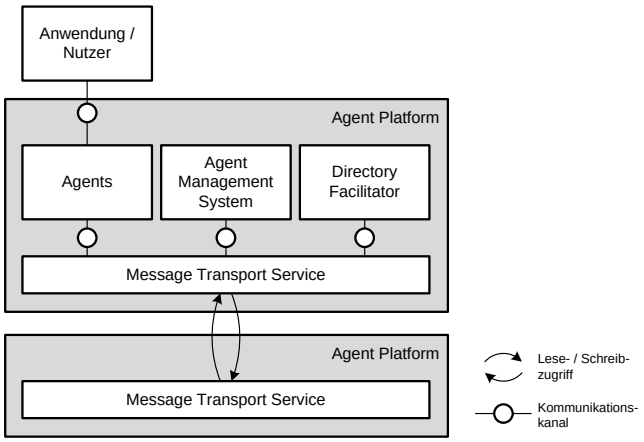


Abbildung 30: FIPA-konformes Agentenmanagement

Die *Agent Platform* stellt die Infrastruktur für die Ausführung von Agenten bereit. Hierzu zählen Systemressourcen ebenso wie weitere FIPA-spezifische Komponenten, zu welchen neben den Agenten selbst auch das *Agent Management System*, der *Directory Facilitator* sowie der *Message Transport Service* zählen (siehe auch Abbildung 30). Der *Directory Facilitator* stellt eine optionale Komponente eines FIPA-konformen Agentensystems dar, welche die Registrierung eines Agenten sowie den durch ihn bereitgestellten Diensten sowie die Suche nach Agenten bzw. Diensten ermöglicht. Das *Agent Management System* kümmert sich um das Lebenszyklusmanagement eines Agenten, im Speziellen dessen Erstellung, Migration und auch Außerbetriebnahme. Der *Message Transport Service* ist für den Austausch von standardisierten Nachrichten auf Basis der FIPA-ACL zwischen den Agenten zuständig.

FIPA standardisiert eine abstrakte Architektur, deren Einhaltung die Kompatibilität verschiedener konkreter Frameworks gewährleistet. Zu den Kernelementen der abstrakten FIPA-Architektur zählen der eben beschriebene *Message Transport Service*, ein *Agent Directory Service*, in welchem Agenten registriert werden müssen, sowie der *Service Directory Service*, an welchen sich Agenten bei der Suche nach benötigten Diensten wenden und ebenfalls angebotene Dienste registrieren können [BCG07].

Das *Java Agent Development Framework* ist eine konkrete Implementierung der durch den FIPA-Standard spezifizierten Inhalte [BCG07]. Agenten, die auf Basis von JADE entwickelt werden, sind autonom und proaktiv. Darüber hinaus verfolgen JADE-Agenten das Paradigma der losen Kopplung, indem sie nachrichtenbasierte asynchrone Kommunikationsverfahren einsetzen. Auf Basis des JADE-Frameworks lassen sich reine Peer-to-Peer-Systeme ebenso wie hybride Ansätze abbilden.[30] Das JADE-Framework bietet darüber hinaus einem Entwickler eine Reihe von Werkzeugen zum Test sowie dem Deployment von agentenbasierten Systemen an.

30 Für eine detaillierte Diskussion der Peer-to-Peer-Technologie (P2P) wird an dieser Stelle auf [SW05] und [Lie08] verwiesen.

A.3 TECHNISCHER VERSUCHSAUFBAU ZUR EVALUATION DER LOKATIONSVERFAHREN

In diesem Abschnitt werden Hintergrundinformationen über die der Evaluation zu Grunde liegende Simulationsumgebung präsentiert.

A.3.1 *Komponenten der AMAS.KOM Workbench*

Gegenstand dieses Abschnitts ist der Versuchsaufbau zur Untersuchung der Lokationsverfahren. Kern des Versuchsaufbaus ist die *AMAS.KOM Workbench*, welche eine umfangreiche Simulationsumgebung zur Untersuchung der Heuristiken bietet. Die *AMAS.KOM Workbench* wurde eigens zu diesem Zweck im Rahmen der vorliegenden Arbeit entwickelt.

Bei der Realisierung der Workbench kommen eine Reihe heterogener Technologien und Produkte zum Einsatz. Mit Ausnahme des *AMAS Model Generator*, welcher auf Basis des Microsoft .NET Frameworks in Version 3.5 entwickelt wurde, sind die weiteren Komponenten Java-basiert. Die Solver-Suite Xpress-MP kommt in der Workbench im 2007 Release zum Einsatz. Als Datenbankmanagementsystem findet ein MySQL-Server in Version 5.1 Verwendung. Das Zusammenspiel der einzelnen Komponenten wird durch Verwendung von Shellskripten auf Grundlage der Microsoft PowerShell in Version 1.0 umgesetzt. Der Datenaustausch erfolgt sowohl unter Verwendung des Dateisystems als auch über das Datenbankmanagementsystem.

Abbildung 31 stellt auf Basis der Modellierungssprache FMC die wichtigsten Komponenten der Simulationsumgebung dar. Nachfolgend sollen die einzelnen Komponenten am Beispiel einer vollständigen Simulation einer Testreihe erläutert werden. Eine Testreihe wird durch ein Skript parametrisiert, welches nacheinander die notwendigen Funktionen der einzelnen Komponenten anspricht. Grundlegende Informationen, auf die das Skript zurückgreift, sind Konfigurationsinformationen für den BRITE-Topologiegenerator sowie Zufallsdaten zur Initialisierung der weiteren Komponenten, die so genannten *Seeds*. In einem ersten Schritt werden aus diesem Grund alle für den weiteren Verlauf benötigten Zufallswerte durch den *SeedGenerator* erzeugt, welcher unter Verwendung der PURAN2-Bibliothek (siehe Abschnitt 3.6.2) diese Werte erzeugt. Nach deren Erzeugung erfolgt der Aufruf von BRITE, welcher die Seeds sowie eine Konfigurationsdatei verwendet, um eine Topologie zu erzeugen. Das Ergebnis wird in einem Verzeichnisbaum zusammen mit dem zugehörigen Seed abgelegt, um eine erneute Durchführung desselben Experiments zu ermöglichen. Das Steuerungsskript ruft in einem nächsten Schritt den *AMAS Model Generator* auf, welcher als Eingabewert die BRITE-Topologie sowie in dem Steuerungsskript beschriebene Konfigurationsparameter benötigt. Ergebnis der Ausführung dieser Komponente ist ein *AMG Model*, welches eine vollständige Experimentbeschreibung bestehend aus der Beschreibung der Kapazitätsrestriktionen, der Topologie als Adjazenzmatrix, der Nachfrage der Dienstanbieter sowie der Installations- und Betriebskosten beinhaltet (siehe Beispiel in Abschnitt A.3.4). Eine solche Modellbeschreibung wird ebenfalls im Verzeichnisbaum archiviert. Die Modellbeschreibung ist Grundlage für den Aufruf des *Xpress-MP* Solvers sowie der Heuristiken. Der Aufruf wird durch

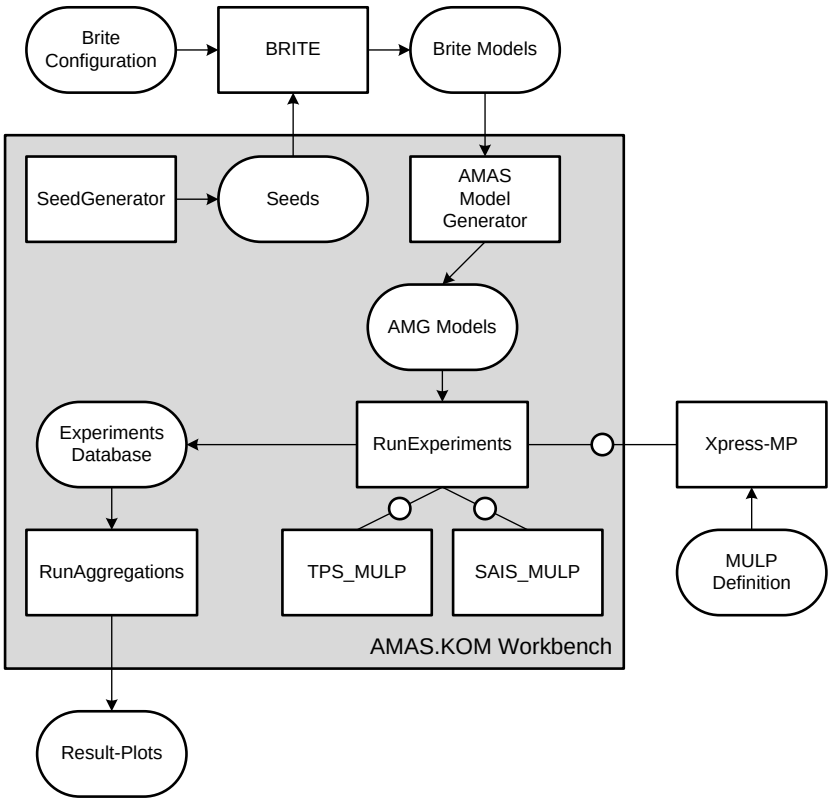


Abbildung 31: Aufbau der AMAS.KOM Workbench

die Komponente *RunExperiments* durchgeführt, welche ebenfalls für die Dokumentation der Ergebnisse im Datenbankmanagementsystem zuständig ist. Hierbei kommt die BeanKeeper-Bibliothek für Objektpersistenz zum Einsatz. Der Solver greift zur Lösung des Modells auf eine gesondert beschriebene Definition des MULP zurück, welche in der Mosel-Sprache das Optimierungsproblem spezifiziert. Abschließend erfolgt nach Sicherung der Ergebnisse der Aufruf der *RunAggregations* Komponente durch das Steuerungsskript, welche die Auswertung der Daten sowie das Erstellen der Diagramme durchführt. Die Ergebnisdokumente werden ebenfalls im Verzeichnisbaum archiviert. Zum Erstellen sowie dem Export der Diagramme kommt die JFree Charting Bibliothek zum Einsatz.

Nicht in Abbildung 31 berücksichtigt ist der *AMAS Results Visualizer*. Das Visualisierungswerkzeug ergänzt die Workbench um eine Möglichkeit zur grafischen Aufbereitung der Topologien sowie der Lösungen der einzelnen Experimente. Das Visualisierungswerkzeug greift hierbei auf das Java Universal Network/Graph Framework zurück.[31]

Die Testreihen, deren Ergebnisse Gegenstand dieser Arbeit sind, wurden auf einem Intel 2 GHz DualCore System mit 2 GB Arbeitsspeicher simuliert. Jede Testserie, welche sämtliche der in Tabelle 7 in Abschnitt 3.6.2 beschriebenen Testreihen ausführt, benötigt mit dieser Systemkonfiguration eine Laufzeit von rund 5 Stunden.

31 http://jung.sourceforge.net/ – abgerufen am 14.04.2009.

A.3.2 *Paket- und Klassenübersicht*

Tabelle 18 zeigt eine Übersicht über die Paket- und Klassenstruktur der wichtigsten Komponenten der AMAS.KOM Workbench.

Paket	**Enthaltene Klassen**
kom.amas.aggregations	CostChartData DataTables JDBCConnector NodeChartData ResultChartGenerator ResultSetAggregator RunAggregations TimeChartData
kom.amas.amg	AmgProgram StrongRnd
kom.amas.foundations	Model ResultSet Helper
kom.amas.heuristics	RndMulp RndTreeMulp SaisMulp TpsMulp
kom.amas.solver	SolverMulp
kom.amas.utils	SeedGenerator
kom.amas.visualizer	FileParser ConfigDialog VisualizerApp

Tabelle 18: Paket- und Klassenübersicht

A.3.3 *Beispielkonfiguration des BRITE-Topologiegenerators*

Das folgende Listing stellt die Konfiguration des BRITE-Topologiegenerators anhand eines Beispiels dar.

```
BriteConfig

BeginModel
        Name =  4  #Router Barabasi=2, AS Barabasi =4
        N = 100    #Number of nodes in graph
        HS = 1000  #Size of main plane (number of squares)
        LS = 100   #Size of inner planes (number of squares)
        NodePlacement = 1 #Random = 1, Heavy Tailed = 2
        m = 2      #Number of nodes each new node connects to
        BWDist = 2 #Constant = 1, Uniform =2,
                   #HeavyTailed = 3, Exponential =4
        BWMin = 10.0
        BWMax = 50.0
EndModel

BeginOutput
        BRITE = 1  #0 = Do not save as BRITE, 1 = save as BRITE
        OTTER = 0  #0 = Do not visualize with Otter, 1 = Visualize
EndOutput
```

Eine BRITE-Konfiguration besteht im Kern aus zwei Blöcken, welche das Modell selbst sowie dessen Ausgabeformate beschreiben. Der Parameter *Name* im Bereich der Modellspezifikation ermöglicht die Konfiguration des Verfahrens zur Topologiegenerierung. Es werden zwei unterschiedliche Ansätze auf Basis des Algorithmus von Barabasi-Albert (siehe Abschnitt 3.6.1) unterstützt. Für eine Konfiguration eines Waxman-basierten Verfahrens kommt eine andere Konfigurationsvorlage zum Einsatz. Die im Beispiel getroffene Auswahl generiert im Folgenden eine Topologie, welche aus autonomen Systemen besteht. Der Parameter *N* definiert die Anzahl der Knoten, die in einer Ebene spezifiziert aus $HS \times LS$ platziert werden. Die Platzierung erfolgt dabei anhand einer durch *NodePlacement* zu selektierenden statistischen Verteilung. Parameter *m* gibt die Konnektivität eines neuen Knotens an. Die Entfernung von Knoten wird ebenfalls stochastisch gesteuert, indem durch den Parameter *BWDist* eine Verteilung ausgewählt wird. Die Parameter *BWMin* und *BWMax* definieren im Sinne von BRITE das Intervall, in welchem sich die zufällig verteilten Entfernungen bewegen. In unserem Modell wird das Intervall zur Begrenzung der Betriebskosten verwendet.

Im Ausgabeteil der BRITE-Konfiguration kann gewählt werden, ob neben dem BRITE-Format selbst noch andere Exportformate zum Einsatz kommen sollen. Das ebenfalls unterstützt OTTER-Format dient der Visualisierung von Topologien.

A.3.4 *Beispiel einer AMG-Experimentspezifikation*

Das nachfolgende Listing zeigt das Beispiel eines mit dem *AMAS Model Generator* erzeugten Experiments. Kernelemente der Experimentspezifikation sind drei Vektoren und eine Matrix, welche die Modellparameter in einem für den Solver verständlichen Format beinhalten.

```
!timestamp: 13.04.2009 20:24:44
! number of overall nodes: 10
! number of service providers: 5
! setup cost ratio for units: 0,25
! number of restricted nodes: 2
! alignment demand of service provider: fixed to 5

CAP: [999 0 999 999 999 999 999 999 0 999]

TOPOCOST: [0 12 33 999 999 999 999 999 54 999
12 0 17 53 29 32 39 11 10 999
33 17 0 14 33 999 999 999 999 51
999 53 14 0 999 42 999 999 999 999
999 29 33 999 0 999 52 42 999 999
999 32 999 42 999 0 999 999 999 999
999 39 999 999 52 999 0 999 999 32
999 11 999 999 42 999 999 0 999 999
54 10 999 999 999 999 999 999 0 999
999 999 51 999 999 999 32 999 999 0]

DEM: [0 0 5 5 5 0 5 0 0 5]

CFIX: [0 11 999 999 999 4 999 6 3 999]
```

Im Detail besteht das Listing zunächst aus einem umfangreichen Header, mit welchem die Experimente leicht identifiziert werden können. Der erste im Listing enthaltene Vektor *CAP* beschreibt mögliche Kapazitätsrestriktionen des Experiments. Der Wert *999* kennzeichnet hierbei unbegrenzte Kapazitäten, der Wert *0* einen Ausschluss des Knotens für die Platzierung von Überwachungs- und Steuerungseinheiten. Die Matrix *TOPOCOST* bildet die durch BRITE generierte Topologie ab und repräsentiert gleichzeitig die Betriebskosten einer Verbindung zwischen zwei Knoten. Der Wert *999* modelliert nicht vorhandene Kanten innerhalb eines sonst vollständig verbundenen Graphen. Der Vektor *DEM* kennzeichnet die Knoten, welche Dienstanbieter repräsentieren. Der Korrekturbedarf ist für alle Dienstanbieter im obigen Beispiel konstant. Der Vektor *CFIX* repräsentiert abschließend die Installationskosten der Einheiten auf den jeweiligen Knoten. Der Wert *999* kennzeichnet ebenfalls die Dienstanbieter in dem gegebenen Szenario, da auf deren Knoten keine Einheit ausgeführt werden darf.

A.4 WEITERE MESSERGEBNISSE

In den nachfolgenden Unterabschnitten werden zusätzliche Ergebnisse sowie andere Darstellungsformen der bereits präsentierten Ergebnisse aufgeführt. Auf ihre Diskussion wird an dieser Stelle verzichtet.

A.4.1 *Übersicht über alle Testreihen*

Gegenstand der Abbildungen 32 bis 38 ist die Darstellung der Ersparnisse durch die Verteilung der Überwachung und Steuerung *(a)*, die Darstellung der Lösungsgüte der Heuristiken *(b)*, die Abbildung der absoluten Laufzeit der einzelnen Verfahren unter Einbezug der Streuung der Messwerte *(c)* sowie das Verhältnis von Einheiten zu Dienstanbietern *(d)*.

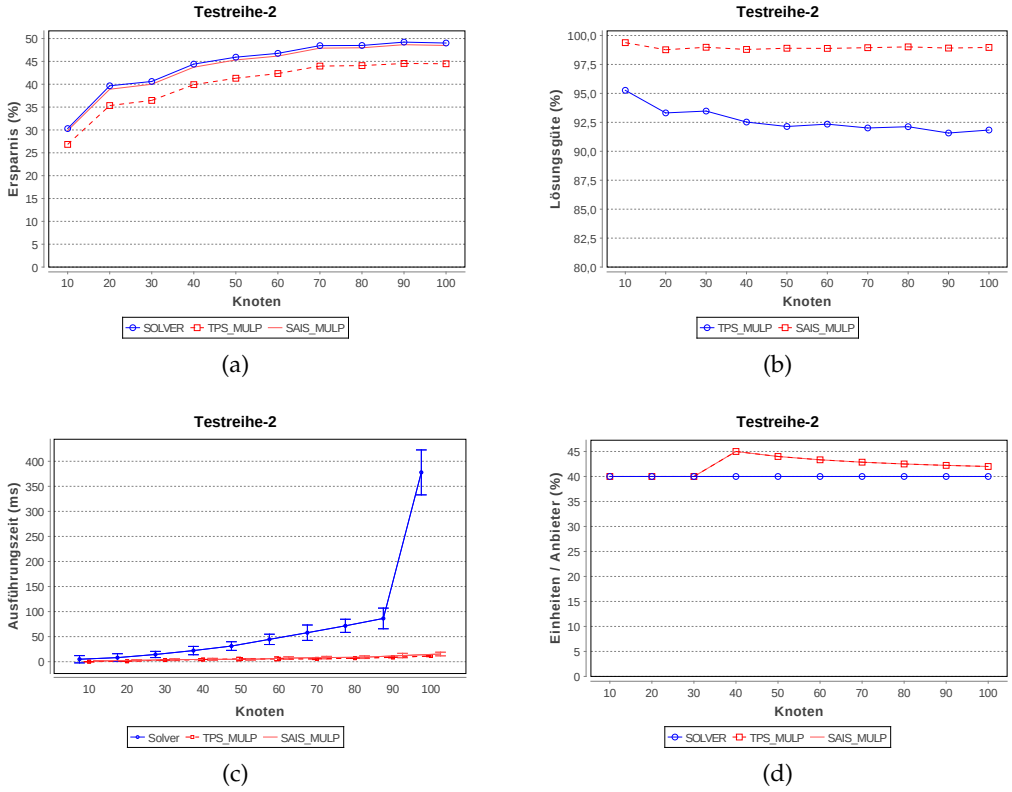
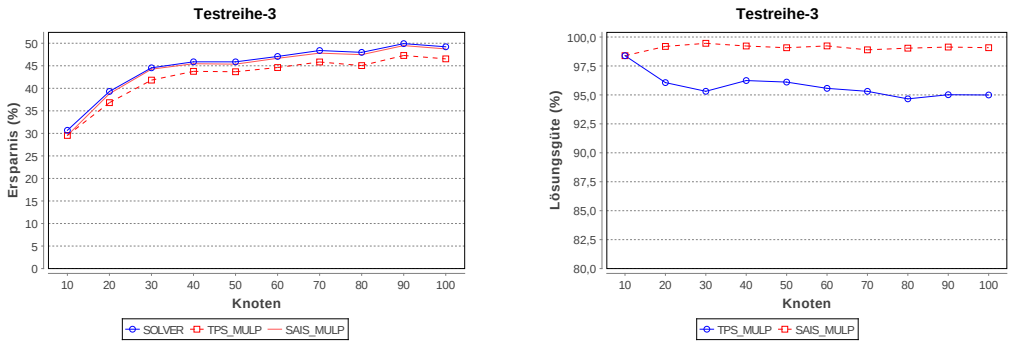
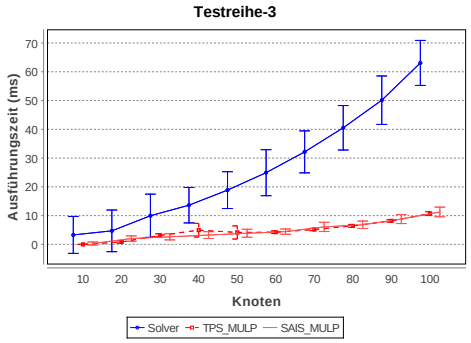


Abbildung 32: Ergebnisse der Testreihe 2

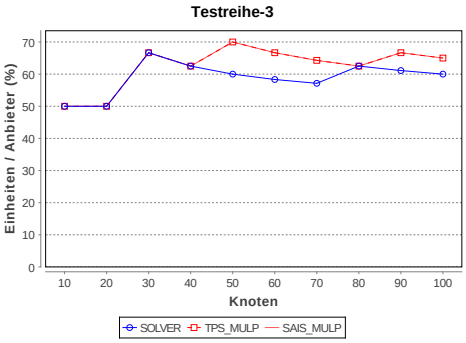


(a)

(b)

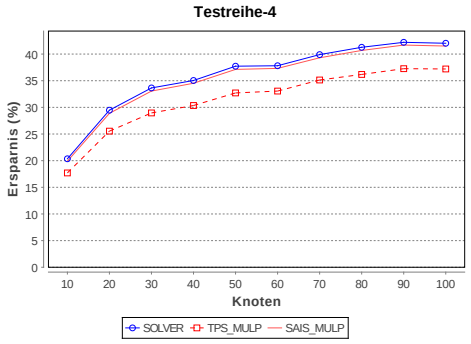


(c)

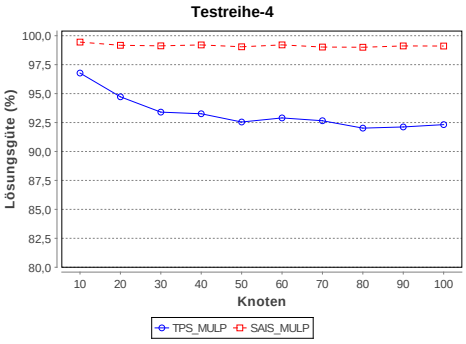


(d)

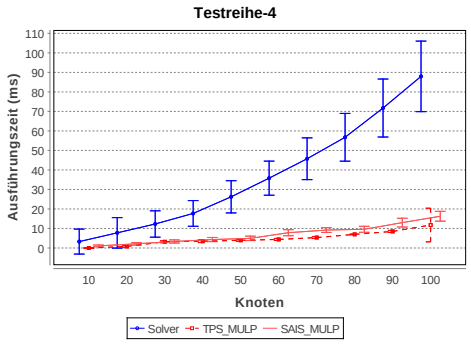
Abbildung 33: Ergebnisse der Testreihe 3



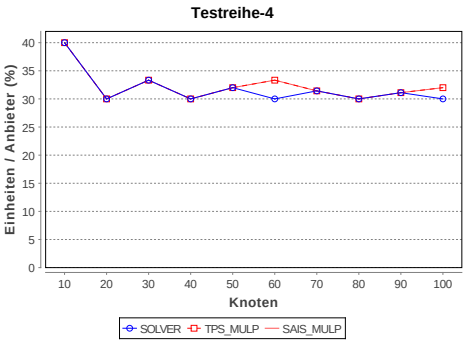
(a)



(b)



(c)



(d)

Abbildung 34: Ergebnisse der Testreihe 4

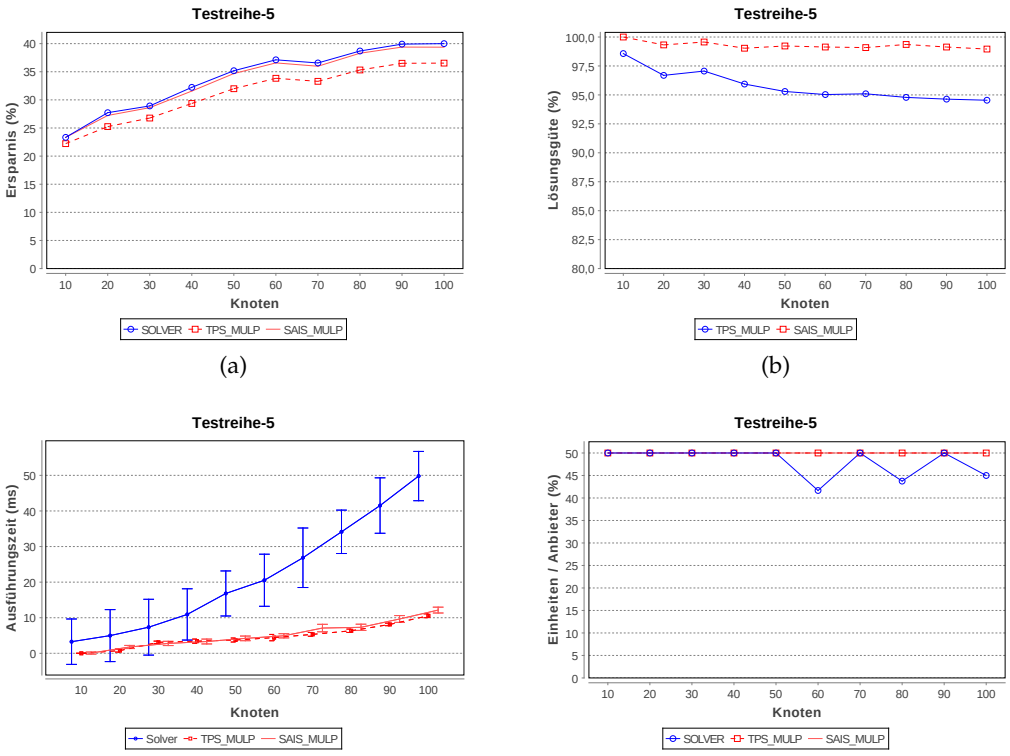


Abbildung 35: Ergebnisse der Testreihe 5

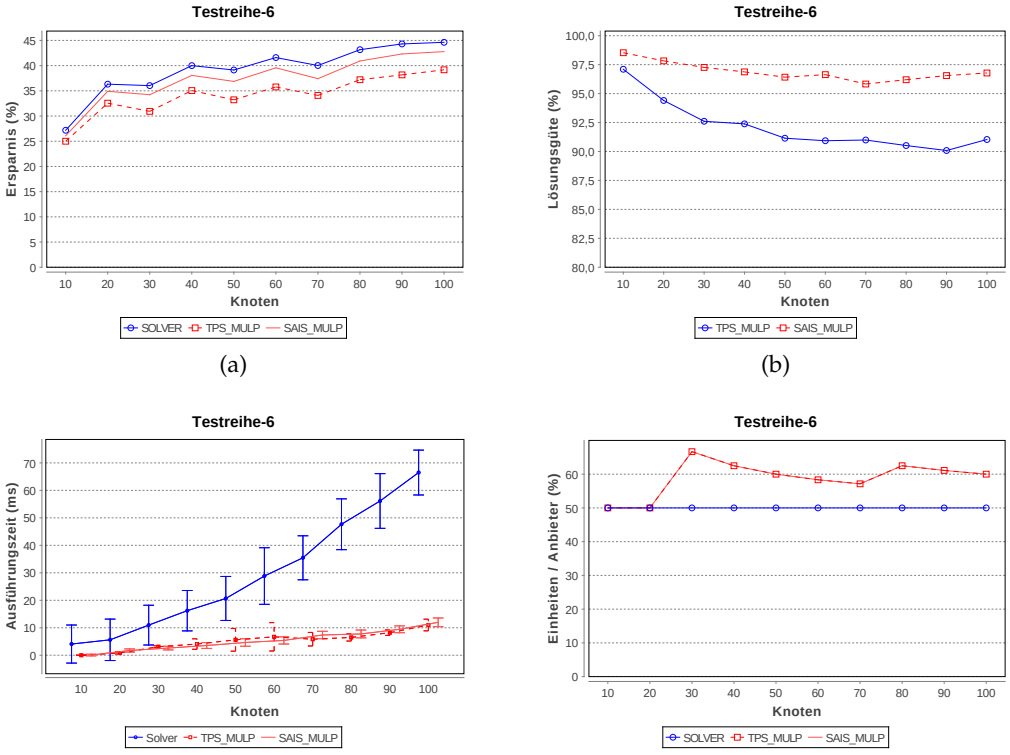
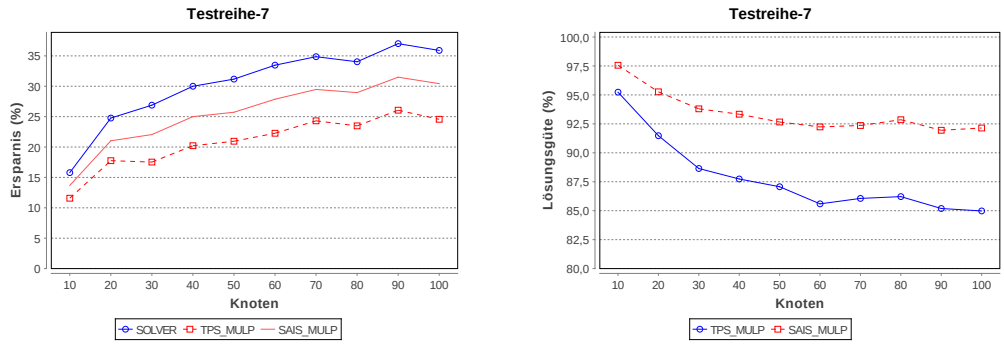
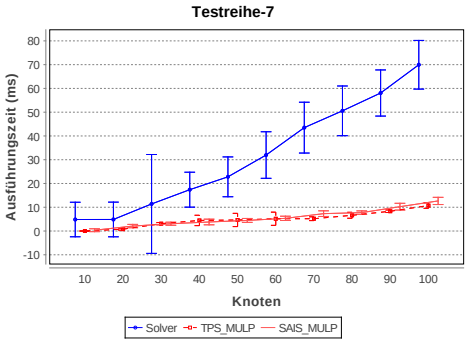


Abbildung 36: Ergebnisse der Testreihe 6

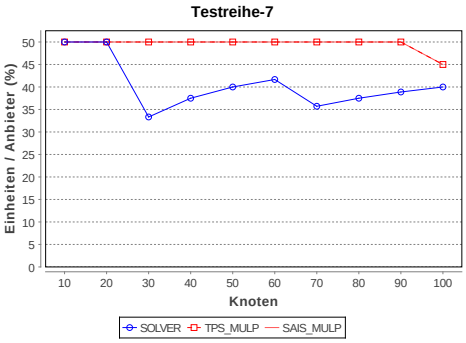


(a)

(b)

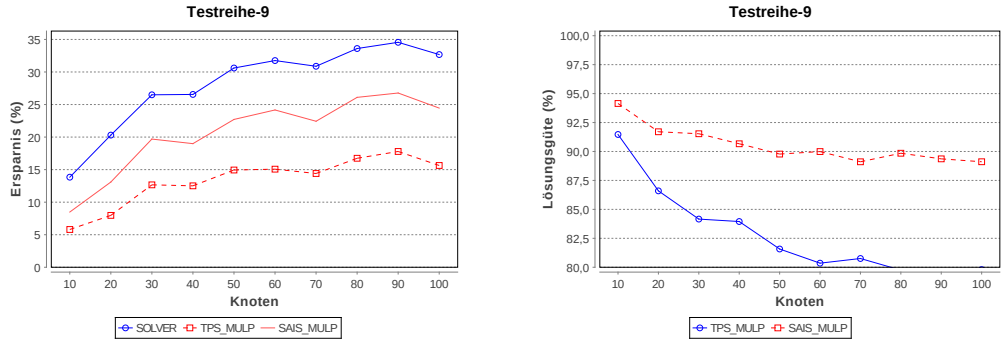


(c)



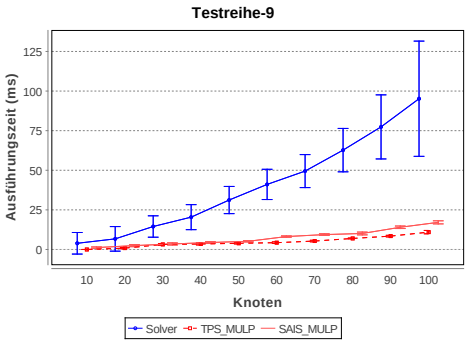
(d)

Abbildung 37: Ergebnisse der Testreihe 7

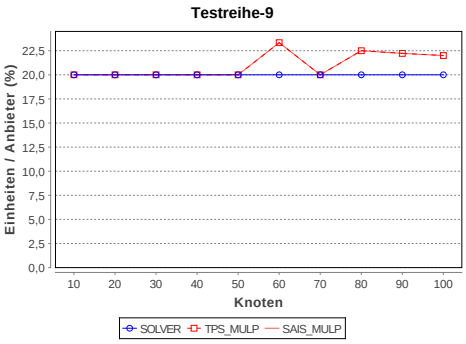


(a)

(b)



(c)



(d)

Abbildung 38: Ergebnisse der Testreihe 9

A.4.2 *Tabellarische Darstellung der Lösungsgüte*

Inhalte der in diesem Abschnitt dargestellten Tabellen sind die Aufgliederung der Lösungsgüte des in dieser Arbeit definierten Eröffnungsverfahrens *TPS_MULP* (siehe Tabelle 20) sowie des angewendeten Verbesserungsverfahrens *SAIS_MULP* (siehe Tabelle 19). Die Spalten der Tabellen kennzeichnen die jeweilige Topologiegröße, wohingegen die Zeilen die einzelnen Testreihen repräsentieren.

	Knoten									
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
1	100,00	100,00	98,88	98,72	99,31	98,85	99,04	98,94	99,04	99,16
2	99,38	98,77	98,98	98,80	98,90	98,89	98,95	99,02	98,91	98,97
3	98,39	99,19	99,46	99,22	99,07	99,23	98,89	99,04	99,13	99,08
4	99,45	99,17	99,12	99,20	99,04	99,21	99,02	99,00	99,12	99,11
5	100,00	99,32	99,57	99,03	99,23	99,14	99,09	99,36	99,14	98,96
6	98,53	97,83	97,26	96,88	96,42	96,64	95,83	96,20	96,56	96,78
7	97,56	95,27	93,80	93,33	92,66	92,24	92,36	92,86	91,95	92,14
8	98,80	94,79	92,64	90,24	89,30	89,12	87,93	87,90	87,91	85,82
9	94,15	91,70	91,53	90,67	89,78	89,99	89,11	89,85	89,36	89,11

Tabelle 19: Lösungsgüte von SAIS_MULP (in Prozent)

	Knoten									
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
1	100,00	96,67	96,20	95,47	95,99	95,56	96,04	95,30	95,40	95,64
2	95,27	93,31	93,47	92,52	92,14	92,34	92,01	92,12	91,58	91,84
3	98,39	96,06	95,31	96,24	96,11	95,57	95,31	94,66	95,02	94,99
4	96,77	94,72	93,40	93,26	92,54	92,90	92,66	92,02	92,12	92,32
5	98,57	96,69	97,06	95,94	95,30	95,03	95,10	94,79	94,64	94,54
6	97,10	94,41	92,61	92,38	91,15	90,93	90,99	90,51	90,08	91,04
7	95,24	91,48	88,64	87,74	87,07	85,59	86,06	86,22	85,19	84,98
8	96,47	93,33	89,35	87,13	86,60	85,35	83,40	84,06	82,70	80,50
9	91,47	86,60	84,16	83,95	81,58	80,35	80,76	79,75	79,60	79,80

Tabelle 20: Lösungsgüte von TPS_MULP (in Prozent)

A.4.3 *Tabellarische Darstellung der relativen Laufzeiten*

Analog zum vorangegangenen Abschnitt werden in diesem Abschnitt weitere Messergebnisse tabellarisch präsentiert. Die in den Tabellen 21 und 22 dargestellten relativen Laufzeiten definieren sich als Verhältnis der Laufzeit einer Heuristik zu der Laufzeit des Solvers. Wie im vorangegangenen Abschnitt kennzeichnen die Spalten der Tabellen die jeweilige Topologiegröße und die Zeilen die einzelnen Testreihen.

	Knoten									
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
1	0,00	40,00	33,33	20,00	14,29	13,33	11,90	12,00	13,79	8,77
2	25,00	37,50	28,57	22,73	16,13	15,91	14,04	12,68	13,95	3,98
3	0,00	50,00	33,33	23,08	22,22	16,67	18,75	17,50	18,00	17,46
4	33,33	28,57	25,00	23,53	19,23	22,86	20,00	17,86	18,31	18,39
5	0,00	50,00	42,86	30,00	25,00	25,00	26,92	20,59	24,39	24,49
6	0,00	40,00	30,00	25,00	25,00	17,86	20,00	17,02	16,07	18,18
7	25,00	50,00	27,27	23,53	18,18	16,13	16,28	16,00	17,24	18,84
8	50,00	33,33	27,27	25,00	22,73	15,15	19,51	16,00	16,67	17,11
9	33,33	50,00	28,57	20,00	16,13	19,51	20,41	16,13	18,18	17,89

Tabelle 21: Relative Laufzeit von SAIS_MULP (in Prozent)

	Knoten									
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
1	0,00	20,00	33,33	26,67	19,05	13,33	14,29	14,00	13,79	9,65
2	0,00	12,50	21,43	18,18	16,13	13,64	10,53	9,86	10,47	2,92
3	0,00	25,00	33,33	38,46	22,22	16,67	15,63	17,50	16,00	17,46
4	0,00	14,29	25,00	17,65	15,38	14,29	11,11	12,50	12,68	13,79
5	0,00	25,00	42,86	30,00	25,00	20,00	19,23	17,65	19,51	22,45
6	0,00	20,00	30,00	25,00	30,00	25,00	17,14	14,89	14,29	16,67
7	0,00	25,00	27,27	23,53	22,73	16,13	11,63	14,00	13,79	15,94
8	0,00	16,67	27,27	18,75	18,18	15,15	12,20	14,00	13,33	14,47
9	0,00	16,67	21,43	15,00	12,90	9,76	10,20	11,29	11,69	11,58

Tabelle 22: Relative Laufzeit von TPS_MULP (in Prozent)

A.5 SPRACHSPEZIFIKATION WS-RE2POLICY

Nachfolgend wird die Sprachspezifikation von WS-Re2Policy in einer formalen Darstellungsweise präsentiert. Dabei wird auf das vom W3C standardisierte XML Schema zurückgegriffen, welches die Spezifikation von XML-basierten Sprachen auf XML-Basis ermöglicht.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created by Nicolas Repp (KOM TUD) -->
<xs:schema xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:xs="http://
    www.w3.org/2001/XMLSchema" xmlns:re2="http://www.kom.tu-darmstadt.de/~nrepp/
    ws/2007/06/ws-re2policy" xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/
    securitypolicy" targetNamespace="http://www.kom.tu-darmstadt.de/~nrepp/ws
    /2007/06/ws-re2policy" elementFormDefault="qualified" blockDefault="#all">
        <xs:import namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"
            schemaLocation="http://schemas.xmlsoap.org/ws/2004/08/addressing"/>
        <xs:import namespace="http://schemas.xmlsoap.org/ws/2004/09/policy"
            schemaLocation="http://schemas.xmlsoap.org/ws/2004/09/policy/ws-
            policy.xsd"/>
        <!-- Requirements & Reaction Suite -->
        <xs:element name="RequirementsReactionsSuite" type="re2:NestedPolicyType
            ">
                <xs:annotation>
                        <xs:documentation xml:lang="en">
                                Testannotation
                        </xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:complexType name="NestedPolicyType">
                <xs:sequence>
                        <xs:element ref="re2:Requirements" maxOccurs="unbounded"
                            />
                        <xs:element ref="re2:Reactions" minOccurs="0" maxOccurs=
                            "unbounded"/>
                        <xs:any namespace="##other" processContents="lax"
                            minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:anyAttribute namespace="##any" processContents="lax"/>
        </xs:complexType>
        <!-- Requirements -->
        <xs:element name="Requirements" type="re2:RequirementsType"/>
        <xs:complexType name="RequirementsType">
                <xs:sequence>
                        <xs:element ref="wsp:Policy"/>
                </xs:sequence>
                <xs:attribute name="RequirementsID" type="xs:integer"/>
                <xs:attribute name="ReactionsNeeded" type="xs:boolean"/>
        </xs:complexType>
        <!-- Reactions -->
        <xs:element name="Reactions" type="re2:ReactionsType"/>
        <xs:complexType name="ReactionsType">
                <xs:sequence>
```

```
                        <xs:any namespace="##targetNamespace" processContents="
                            lax" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="RequirementsID" type="xs:integer"/>
        </xs:complexType>
        <xs:element name="RestartService"/>
        <xs:element name="RenegotiateSLA"/>
        <xs:element name="ReplanExecution"/>
        <xs:element name="SelectDifferentService"/>
        <xs:element name="ReportResults"/>
        <xs:element name="DelegateControl"/>
        <xs:element name="StopExecution"/>
        <xs:element name="Sleep" type="re2:sleeperType"/>
        <xs:element name="AutomatedHandling"/>
        <xs:element name="Iterate" type="re2:IteratorType"/>
        <xs:complexType name="IteratorType">
                <xs:sequence>
                        <xs:any namespace="##targetNamespace" processContents="
                            lax" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="time" type="xs:double"/>
                <xs:attribute name="count" type="xs:nonNegativeInteger"/>
        </xs:complexType>
        <xs:complexType name="sleeperType">
                <xs:attribute name="time" type="xs:double"/>
        </xs:complexType>
</xs:schema>
```

A.6 ERGÄNZUNGEN ZUM AMAS.KOM FRAMEWORK

Im Fokus dieses Abschnitts stehen ergänzende Informationen zur Untersuchung der Performanz des Prototyps sowie zum Entwurf des AMAS.KOM Frameworks.

A.6.1 *Stichprobenumfänge der Performanzmessung*

Die in Abschnitt 5.5 vorgestellten Ergebnisse der Perfomanzmessung des auf dem AMAS.KOM Framework basierenden Prototyps basieren auf der Nutzung des JMeter Messwerkzeugs. In den Experimenten werden durch JMeter in 10 Minuten so viele Dienstaufrufe pro Thread-Klasse wie möglich durchgeführt. Zur Erläuterung der Messergebnisse in Abschnitt 5.5 wird in Tabelle 23 die jeweilige Anzahl der Messungen pro Testfall dargestellt.

Threads	**5**	**10**	**15**	**20**	**25**	**30**
Ohne AMAS	3.242	3.059	3.064	3.027	2.980	2.948
Mit AMAS	2.765	2.799	2.683	2.590	2.621	2.608

Tabelle 23: Anzahl der Proben pro Messung

A.6.2 *Interaktionen der Komponenten als Sequenzdiagramm*

Als Ergänzung zum Entwurf des AMAS.KOM Frameworks sowie der in Abschnitt 5.3.2 dargestellten Interaktionen zwischen den Komponenten des Frameworks werden in Abbildung 39 in Form eines UML-Sequenzdiagramms die Interaktionen zwischen den Komponenten des Frameworks visualisiert.

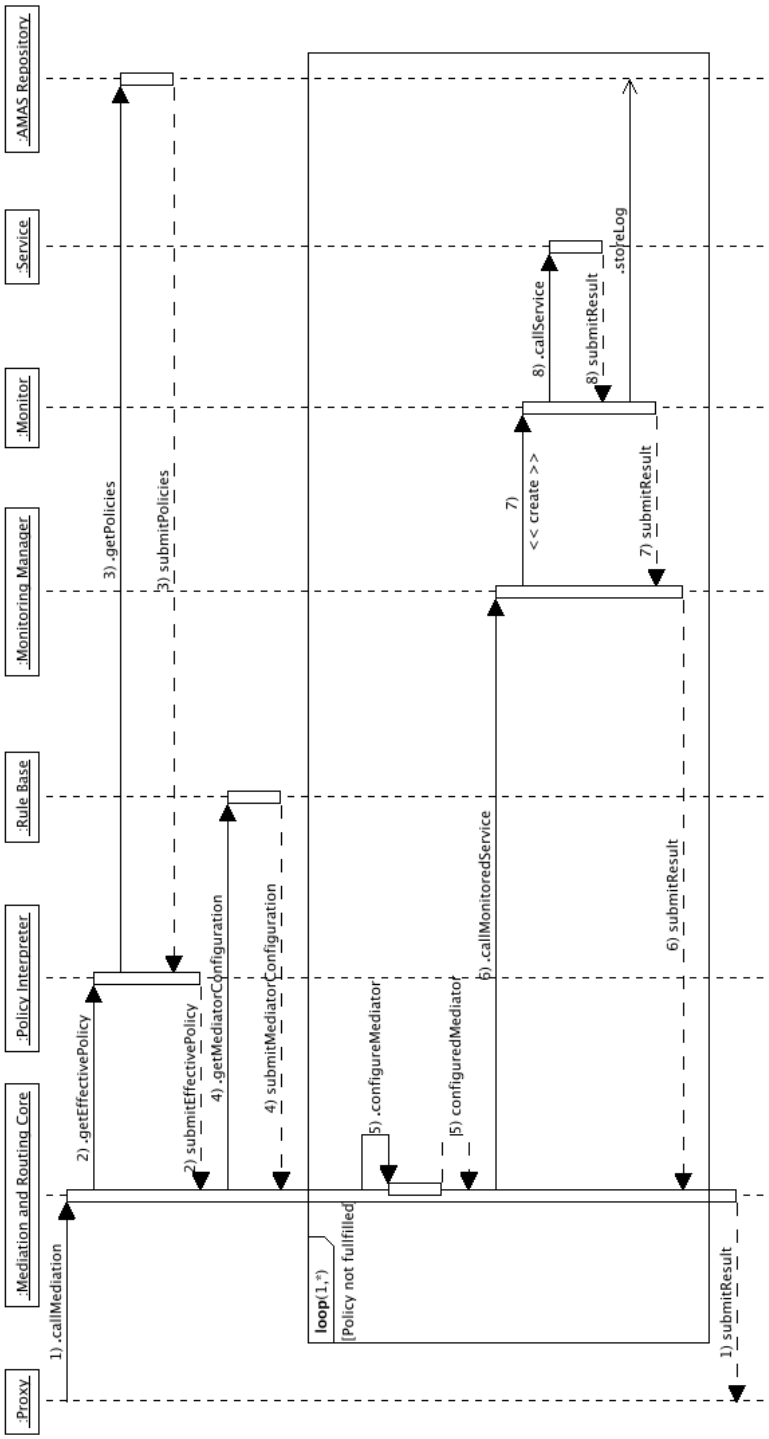


Abbildung 39: Sequenzdiagramm AMAS.KOM

B

PUBLIKATIONEN DES AUTORS

B.1 HAUPTVERÖFFENTLICHUNGEN

[1] Repp, Nicolas: mobileISM – Entwicklung eines Frameworks zur benutzerunterstützenden Verwaltung von Diensten auf mobilen Endgeräten. In: *Wirtschaftsinformatik* 45 (2003), Dezember, Nr. 6, S. 658–661

[2] Repp, Nicolas: Monitoring of Services in Distributed Workflows. In: *Tagungsband 3rd International Conference on Software and Data Technologies*, Juli 2008, S. 15–25

[3] Repp, Nicolas ; Berbner, Rainer ; Eckert, Julian ; Heckmann, Oliver ; Schulte, Stefan ; Steinmetz, Ralf: Der Einfluß von Transportschicht-Anomalien auf die Performanz von Web Services. In: *Tagungsband 5. Fachtagung Kommunikation in Verteilten Systemen*, Februar 2007, S. 117–122

[4] Repp, Nicolas ; Berbner, Rainer ; Heckmann, Oliver ; Steinmetz, Ralf: A Cross-Layer Approach to Performance Monitoring of Web Services. In: *Tagungsband Workshop on Emerging Web Services Technology at the 4th European Conference on Web Services*, 2006, S. 21–32

[5] Repp, Nicolas ; Berbner, Rainer ; Lenz, Jörg ; Kaplan, Christiane ; Perez, Alejandro ; Heckmann, Oliver ; Steinmetz, Ralf: Digitalisierte eigenhändige Unterschrift im Online-Banking. In: *Tagungsband D-A-CH Security*, März 2006, S. 381–391

[6] Repp, Nicolas ; Berbner, Rainer ; Steinmetz, Ralf: Integration von Vertriebskanälen durch einheitliche Authentifizierungs- und Autorisierungsverfahren. In: *Zeitschrift für das gesamte Kreditwesen* (2007), September, Nr. 3-2007, S. 16–18

[7] Repp, Nicolas ; Eckert, Julian ; Schulte, Stefan ; Niemann, Michael ; Berbner, Rainer ; Steinmetz, Ralf: Towards Automated Monitoring and Alignment of Service-based Workflows. In: *Tagungsband IEEE International Conference on Digital Ecosystems and Technologies*, Februar 2008, S. 235–240

[8] Repp, Nicolas ; Heinemann, Elisabeth ; Ortner, Erich: mobileISM – Ein Framework zu adaptiven Unterstützung der Nutzung mobiler Dienste. In: *Tagungsband 11. GI-Workshop Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen*, Oktober 2003

[9] Repp, Nicolas ; Hudert, Sebastian ; Bleul, Steffen: PIK Special Edition on Service-Oriented Computing (Editorial). In: *PIK – Praxis der Informationsverarbeitung und Kommunikation* 31 (2008), Dezember, Nr. 4, S. 208–210

[10] Repp, Nicolas ; Martin, Wolfgang: *SOA Check 2008: Status Quo und Trends im Vergleich zum SOA Check 2007*. IT-Verlag, Sauerlach, April 2008

[11] Repp, Nicolas ; Mauthe, Andreas U. ; Steinmetz, Ralf: Chancen und Risiken bei der Einführung von IT-Governance-Frameworks. In: *IT-Governance* 2 (2008), März, Nr. 3, S. 9–14

[12] Repp, Nicolas ; Miede, André ; Niemann, Michael ; Steinmetz, Ralf: WS-Re2Policy: A policy language for distributed SLA monitoring and enforcement. In: *Tagungsband 3rd International Conference on Systems and Networks Communications*, Oktober 2008, S. 256–261

[13] Repp, Nicolas ; Schuller, Dieter ; Siebenhaar, Melanie ; Miede, André ; Niemann, Michael ; Steinmetz, Ralf: On distributed SLA monitoring and enforcement in service-oriented systems. In: *International Journal On Advances in Systems and Measurements* 2 (2009), Mai, Nr. 1, S. 33–43

[14] Repp, Nicolas ; Schulte, Stefan ; Eckert, Julian ; Berbner, Rainer ; Steinmetz, Ralf: An Approach to the Analysis and Evaluation of an Enterprise Service Ecosystem. In: *Tagungsband Workshop on Architectures, Concepts and Technologies for Service Oriented Computing at the 2nd International Conference on Software and Data Technologies*, Juli 2007, S. 42–51

[15] Repp, Nicolas ; Schulte, Stefan ; Eckert, Julian ; Berbner, Rainer ; Steinmetz, Ralf: Service-Inventur: Aufnahme und Bewertung eines Services-Bestands. In: *Tagungsband Workshop MDD, SOA und IT-Management*, April 2007, S. 13–22

B.2 MITAUTORENSCHAFT UND SONSTIGE VERÖFFENTLICHUNGEN

[1] Berbner, Rainer ; Grollius, Tobias ; Repp, Nicolas ; Eckert, Julian ; Heckmann, Oliver ; Ortner, Erich ; Steinmetz, Ralf: Management of Service-oriented Architecture (SOA)-based Application Systems. In: *Enterprise Modelling and Information Systems Architectures* 1 (2007), Mai, Nr. 2, S. 14–26

[2] Berbner, Rainer ; Grollius, Tobias ; Repp, Nicolas ; Heckmann, Oliver ; Ortner, Erich ; Steinmetz, Ralf: An approach for the Management of Service-oriented Architecture (SOA) based Application Systems. In: *Tagungsband Workshop Enterprise Modelling and Information Systems Architectures*, Oktober 2005

[3] Berbner, Rainer ; Spahn, Michael ; Repp, Nicolas ; Heckmann, Oliver ; Steinmetz, Ralf: An Approach for Replanning of Web Service Workflows. In: *Tagungsband 13th Americas Conference on Information Systems*, August 2006

[4] Berbner, Rainer ; Spahn, Michael ; Repp, Nicolas ; Heckmann, Oliver ; Steinmetz, Ralf: Heuristics for QoS-aware Web Service Composition. In: *Tagungsband 4th IEEE International Conference on Web Services*, September 2006

[5] Berbner, Rainer ; Spahn, Michael ; Repp, Nicolas ; Heckmann, Oliver ; Steinmetz, Ralf: Dynamic Replanning of Web Service Workflows. In: *Tagungsband IEEE International Conference on Digital Ecosystems and Technologies*, IEEE, Februar 2007

[6] Berbner, Rainer ; Spahn, Michael ; Repp, Nicolas ; Heckmann, Oliver ; Steinmetz, Ralf: WSQoSX – A QoS architecture for Web Service workflows. In: *Tagungsband 5th International Conference on Service-Oriented Computing*, September 2007

[7] Eckert, Julian ; Bachhuber, Marc ; Repp, Nicolas ; Steinmetz., Ralf: The Implementation of Service-oriented Architectures in the German Banking Industry – A Case Study. In: *Tagungsband 15th Americas Conference on Information Systems*, August 2009

[8] Eckert, Julian ; Ertogrul, Deniz ; Miede, André ; Repp, Nicolas ; Steinmetz, Ralf: Resource Planning Heuristics for Service-oriented Workflows. In: *Tagungsband IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, Dezember 2008, S. 591–597

[9] Eckert, Julian ; Ertogrul, Deniz ; Papageorgiou, Apostolos ; Repp, Nicolas ; Steinmetz, Ralf: The Impact of Service Pricing Models on Service Selection. In: *Tagungsband 4th International Conference on Internet and Web Applications and Services*, Mai 2009

[10] Eckert, Julian ; Pandit, Krishna ; Repp, Nicolas ; Berbner, Rainer ; Steinmetz, Ralf: Worst-Case Performance Analysis of Web Service Workflows. In: *Tagungsband 9th International Conference on Information Integration and Web-based Application & Services*, Dezember 2007

[11] Eckert, Julian ; Repp, Nicolas ; Niemann, Michael ; Billeb, Marc ; Schäfer, Achim ; Steinmetz, Ralf: IT Organization as a Limiting Factor for the Success of Service-Oriented Architectures. In: *EFL Quarterly* (2008), April, S. 4–5

[12] Eckert, Julian ; Repp, Nicolas ; Schulte, Stefan ; Berbner, Rainer ; Steinmetz, Ralf: An Approach for Capacity Planning for Web Service Workflows. In: *Tagungsband 13th Americas Conference on Information Systems*, August 2007

[13] Eckert, Julian ; Schulte, Stefan ; Niemann, Michael ; Repp, Nicolas ; Steinmetz, Ralf: Worst-Case Workflow Performance Optimization. In: *Tagungsband 3rd International Conference on Internet and Web Applications and Services*, Juni 2008

[14] Eckert, Julian ; Schulte, Stefan ; Repp, Nicolas ; Berbner, Rainer ; Steinmetz, Ralf: Queuing-based Capacity Planning Approach for Web Service Workflows Using Optimization Algorithms. In: *Tagungsband IEEE International Conference on Digital Ecosystems and Technologies 2008*, Februar 2008

[15] Janiesch, Christian ; Niemann, Michael ; Repp, Nicolas: Governance in the Internet of Services: Governing Service Delivery of Service Brokers. In: *Tagungsband Pre-ICIS Workshop SIG Services*, Dezember 2008

[16] Janiesch, Christian ; Niemann, Michael ; Repp, Nicolas: Towards a Service Governance Framework for the Internet of Services. In: *Tagungsband 17th European Conference on Information Systems*, Juni 2009

[17] Miede, André ; Behuet, Jean-Baptiste ; Papageorgiou, Apostolos ; Niemann, Michael ; Repp, Nicolas ; Steinmetz, Ralf: Qualitative and Quantitative Aspects of Cooperation Mechanisms for Monitoring in Service-oriented Architectures. In: *Tagungsband 3rd IEEE International Conference on Digital Ecosystems and Technologies*, Juni 2009

[18] Miede, André ; Behuet, Jean-Baptiste ; Repp, Nicolas ; Eckert, Julian ; Steinmetz, Ralf: Cooperation Mechanisms for Monitoring Agents in Service-oriented Architectures. In: *Tagungsband der 9. internationalen Tagung Wirtschaftsinformatik*, Februar 2009

[19] Miede, André ; Niemann, Michael ; Schulte, Stefan ; Eckert, Julian ; Kabzeva, Aneta ; Repp, Nicolas ; Steinmetz, Ralf: Selected Topics in Service Engineering and Management for Enterprise Systems. In: *Tagungsband Pre-ICIS 2008 Workshop on Enterprise Systems Research in MIS*, Dezember 2008

[20] Miede, André ; Repp, Nicolas ; Eckert, Julian ; Steinmetz, Ralf: Self-Organization Mechanisms for Information Systems – A Survey. In: *Tagungsband 14th Americas Conference on Information Systems*, August 2008

[21] Miede, André ; Repp, Nicolas ; Steinmetz, Ralf: Concepts of Self-Organization for Service-Oriented Architectures / Technische Universität Darmstadt. Januar 2008 (TR-2008-01). – Forschungsbericht

[22] Niemann, Michael ; Appel, Michael ; Repp, Nicolas ; Steinmetz, Ralf: Towards a Consistent Lifecycle Model in Service Governance. In: *Tagungsband 15th Americas Conference on Information Systems*, August 2009

[23] Niemann, Michael ; Eckert, Julian ; Repp, Nicolas ; Steinmetz, Ralf: SOA Governance: Survey and Model. In: *Tagungsband Pre-ICIS Workshop SIG Services*, Dezember 2008

[24] Niemann, Michael ; Eckert, Julian ; Repp, Nicolas ; Steinmetz, Ralf: Towards a Generic Governance Model for Service-oriented Architectures. In: *Tagungsband 14th Americas Conference on Information Systems*, August 2008

[25] Niemann, Michael ; Janiesch, Christian ; Repp, Nicolas ; Steinmetz, Ralf: Challenges of Governance Approaches for Service-Oriented Architectures. In: *Tagungsband 3rd IEEE International Conference on Digital Ecosystems and Technologies*, Juni 2009

[26] Niemann, Michael ; Siebenhaar, Melanie ; Eckert, Julian ; Schulte, Stefan ; Repp, Nicolas ; Steinmetz, Ralf: SOA in Practice / Technische Universität Darmstadt. 2008 (TR-2008-04). – Forschungsbericht

[27] Papageorgiou, Apostolos ; Schulte, Stefan ; Schuller, Dieter ; Niemann, Michael ; Repp, Nicolas ; Steinmetz, Ralf: Governance of a Service-Oriented Architecture for Environmental and Public Security. In: *Tagungsband 4th International ICSC Symposium Information Technologies in Environmental Engineering*, Mai 2009

[28] SCHULLER, Dieter ; PAPAGEORGIOU, Apostolos ; SCHULTE, Stefan ; ECKERT, Julian ; REPP, Nicolas ; STEINMETZ, Ralf: Process Reliability in Service-Oriented Architectures. In: *Tagungsband 3rd IEEE International Conference on Digital Ecosystems and Technologies*, Juni 2009

[29] SCHULTE, Stefan ; ECKERT, Julian ; REPP, Nicolas ; STEINMETZ, Ralf: An Approach to Evaluate and Enhance the Retrieval of Semantic Web Services. In: *Tagungsband 5th International Conference on Service Systems and Service Management*, Juni 2008, S. 237–243

[30] SCHULTE, Stefan ; KADNER, Kay ; REPP, Nicolas ; STEINMETZ, Ralf: Applied Service Engineering for Single Services and Corresponding Service Landscapes. In: *Tagungsband 15th Americas Conference on Information Systems*, August 2009

[31] SCHULTE, Stefan ; REPP, Nicolas ; BERBNER, Rainer ; STEINMETZ, Ralf ; SCHAARSCHMIDT, Ralf: Service-Oriented Architecture Paradigm: Major Trend or Hype for the German Banking Industry? In: *Tagungsband 13th Americas Conference on Information Systems*, August 2007

[32] SCHULTE, Stefan ; REPP, Nicolas ; ECKERT, Julian ; BERBNER, Rainer ; BLANCKENBURG, Korbinian von ; SCHAARSCHMIDT, Ralf ; STEINMETZ, Ralf: General Requirements of Banks on IT Architectures and the Service-Oriented Architecture Paradigm. In: *Tagungsband 3rd International Workshop FinanceCom at the International Conference on Information Systems*, April 2008, S. 66–80

[33] SCHULTE, Stefan ; REPP, Nicolas ; ECKERT, Julian ; BERBNER, Rainer ; BLANCKENBURG, Korbinian von ; SCHAARSCHMIDT, Ralf ; STEINMETZ, Ralf: Potential Risks and Benefits of Service-Oriented Collaboration – Basic Considerations and Results from an Empirical Study. In: *Tagungsband IEEE International Conference on Digital Ecosystems and Technologies*, Februar 2008

[34] SCHULTE, Stefan ; REPP, Nicolas ; ECKERT, Julian ; BERBNER, Rainer ; STEINMETZ, Ralf ; SCHAARSCHMIDT, Ralf: Familiarity with and Usage of Service-oriented Architectures in German Banks. In: *EFL Quarterly* (2007), April

[35] SCHULTE, Stefan ; REPP, Nicolas ; ECKERT, Julian ; STEINMETZ, Ralf ; BLANCKENBURG, Korbinian von ; SCHAARSCHMIDT, Ralf: Service-oriented Architectures – Status Quo and Prospects for the German Banking Industry. In: *Interoperability in Business Information Systems* 3 (2008), Juli, Nr. 2, S. 9–31

[36] SCHULTE, Stefan ; REPP, Nicolas ; SCHAARSCHMIDT, Ralf ; ECKERT, Julian ; BERBNER, Rainer ; STEINMETZ, Ralf: Potentielle Auswirkungen des Paradigmas der Service-orientierten Architekturen auf die Softwarebranche – Ergebnisse einer Studie aus der deutschen Bankenindustrie. In: *Tagungsband des Stuttgarter Softwaretechnik Forum*, November 2007, S. 21–30

[37] SCHULTE, Stefan ; REPP, Nicolas ; SCHAARSCHMIDT, Ralf ; ECKERT, Julian ; BERBNER, Rainer ; STEINMETZ, Ralf ; BLANCKENBURG, Korbinian von: *Service-orientierte Architekturen – Status quo und Perspektive für die deutsche Bankenbranche*. Books on Demand, November 2007

C

LEBENSLAUF DES VERFASSERS

PERSÖNLICHE DATEN

Nicolas Repp

Geboren am 2. Dezember 1977 in Lich, Hessen
Staatsangehörigkeit deutsch
Familienstand verheiratet

AKADEMISCHER WERDEGANG

seit 4/2008	Technische Universität Darmstadt Doktorand am Fachbereich Elektrotechnik und Informationstechnik
10/1998–11/2003	Technische Universität Darmstadt Studium zum Diplom-Wirtschaftsinformatiker
8/1994–6/1997	Laubach Kolleg der Evangelischen Kirche in Hessen und Nassau Erlangung der allgemeinen Hochschulreife

BERUFSERFAHRUNG

seit 8/2005	Technische Universität Darmstadt Wissenschaftlicher Mitarbeiter und Leiter der Forschungsgruppe IT-Architekturen am Fachgebiet Multimedia Kommunikation
seit 8/2007	Hessisches Telemedia Technologie Kompetenz-Center, Darmstadt Leiter des SOA Competence Center
12/2003–7/2005	PricewaterhouseCoopers, Frankfurt am Main Associate im Bereich Process Assurance
2/2001–10/2003	Technische Universität Darmstadt Studentischer Mitarbeiter am Fachgebiet Entwicklung von Anwendungssystemen / Wirtschaftsinformatik I
9/2002–3/2003	T-Systems Nova, Darmstadt Projektleitung eines Softwareentwicklungsprojekts
7/2001–9/2001	PricewaterhouseCoopers, Frankfurt am Main Praktikum im Bereich Global Risk Management Solutions
8/1998–9/1998	Bosch Telecom, Frankfurt am Main Praktikum in einer Entwicklungsabteilung für Telefonanlagen und Anlagensoftware
8/1997–8/1998	Naturpark Hoher Vogelsberg, Schotten Ableistung des Zivildiensts

LEHRTÄTIGKEIT

seit SS 2009	Hochschule für Technik und Wirtschaft des Saarlandes Vorlesung und Übung „Geschäftsprozessmanagement“ sowie „Systeme und Architekturen“
seit SS 2007	Technische Universität Darmstadt Projektseminar „Communication Systems Lab III: Advanced Topics in Distributed Systems“
seit SS 2006	Lancaster University, Großbritannien Themenstellung und Co-Betreuung von bisher 6 Masterarbeiten an der Lancaster University Management School
SS 2006	Hochschule Liechtenstein, Vaduz Vorlesung und Übung „Softwaretest und -qualitätssicherung“ am Institut für Wirtschaftsinformatik
seit WS 2005/06	Technische Universität Darmstadt Vorlesungsbetreuung und Übung „Communication Networks II“
seit WS 2005/06	Technische Universität Darmstadt Seminar „Communication Systems and Multimedia I: Advanced Topics of Future Internet Research“
seit WS 2005/06	Technische Universität Darmstadt Betreuung von bisher 12 Studien-, Bachelor-, Diplom- und Masterarbeiten aus den Bereichen Wirtschaftsinformatik, Informatik, Wirtschaftsingenieurwesen sowie Informationstechnik
WS 2005/06	Hochschule Liechtenstein, Vaduz Vorlesung und Übung „XML Web Services“ am Institut für Wirtschaftsinformatik

WEITERE AKTIVITÄTEN IM WISSENSCHAFTLICHEN UMFELD

12/2008	Herausgeber einer Sonderausgabe der Zeitschrift „Praxis der Informationsverarbeitung und Kommunikation“ zum Thema „Service-oriented Computing“
3/2009	Organisation und Leitung Workshop „Service-oriented Computing“ auf der 16. GI/ITG-Fachtagung Kommunikation in verteilten Systemen
6/2009	Organisation und Leitung Konferenztrack „Governance in Large Heterogeneous IT Systems“ auf der IEEE Conference on Digital Ecosystems and Technologies
8/2009	Organisation und Leitung Konferenztrack „Enterprise Systems Governance“ auf der 15. Americas Conference on Information Systems
9/2009	Organisation und Leitung Workshop „IT-Governance in verteilten Systemen“ auf der Jahrestagung der Gesellschaft für Informatik

seit 8/2005	Programmkomiteemitgliedschaft und Gutachtertätigkeit für zahlreiche nationale und internationale Konferenzen und Workshops

AUSZEICHNUNGEN

10/2008	Best Paper Award auf der International Conference on Systems and Networks Communications 2008 in Sliema, Malta, für den Beitrag *WS-Re2Policy: A Policy Language for Distributed SLA Monitoring and Enforcement*
8/2008	Best Paper Award auf der Americas Conference on Information Systems 2008 in Toronto, Kanada, für den Beitrag *Towards a Generic Governance Model for Service-oriented Architectures*
2/2008	Zwei Best Presentation Awards auf der International Conference on Digital Ecosystems and Technologies 2008 in Phitsanoluk, Thailand, für die Beiträge *Towards Automated Monitoring and Alignment of Service-based Workflows* und *Queuing-based Capacity Planning Approach for Web Service Workflows Using Optimization Algorithms*

MITGLIEDSCHAFTEN

GI	Gesellschaft für Informatik
ACM	Association for Computing Machinery
AIS	Association for Information Systems
EuroSys	European Professional Society on Computer Systems

D

ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe.

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt 2009